

INSTITUTUL POLITEHNIC IAȘI
Facultatea de Automatică și Calculatoare

Masa

**LUCRĂRILE
CELUI DE AL III-LEA SIMPOZION**

**„STRUCTURI, ALGORITMI
ȘI ECHIPAMENTE DE CONDUCERE
A PROCESELOR INDUSTRIALE**

Vol. II

**IAȘI
25-26 octombrie 1991**

LUCRĂRILE
celui de al III-lea Simpozion

„Structuri, algoritmi și echipamente
de conducere a proceselor
industriale”

Secția D:

Structuri hardware pentru
prelucrarea informațiilor

Secția E:

Structuri software pentru
conducere, simulare și
proiectare asistată

IAȘI
25 - 26 octombrie 1991

Meaga

Controler de linie telefonică

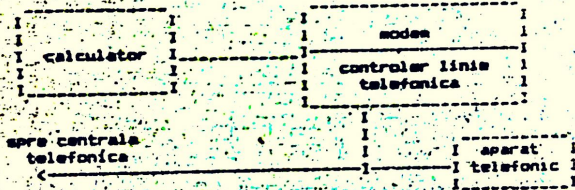
Autorii: conf. dr. ing. Graur Aurian
as. ing. Potorac Dan-Alin
as. ing. Morosan Tudor

Universitatea "Stefan cel Mare" Suceava

1. Introducere

Concesarea acestui echipament a fost determinată de necesitatea realizării de transferuri de date între sistemele de calcul folosind suporturi fizice existente, respectiv linii telefonice. În prezent legătura între doi abonati telefonici pentru transmiterea de date se realizează manual sau se folosesc linii telefonice concesionate.

Aplicația specifică este sugerată în figura de mai jos.



2. Descriere generală

Controlerul de linie telefonică care face obiectul acestei descrieri este destinat stabilirii unei legături pentru comunicațiile seriale prin centrala telefonică automată.

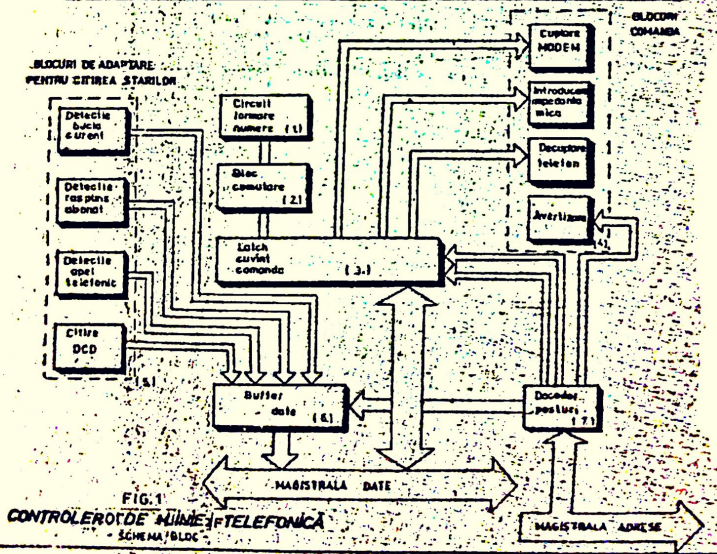
Concret, acest echipament are următoarele funcțiuni:

- sesizează cererea de stabilire a legăturii
- avertizează, un timp prestabilit, ca urmează să se prezinte controlul liniei telefonice
- decuplează un eventual telefon legat la linie
- introduce pe linie o impedanță mare (corespunzătoare stării "telefon închis") sau o impedanță mică (corespunzătoare

- starii "telefon deschis"
- sesizeaza existenta buclei de curent prin linie (sa informati asupra integritatii liniei)
 - formeaza numarul de telefon al abonatului care trebuie apelat
 - cupleaza modemul la linia telefonica
 - detecteaza raspunsul abonatului chemat
 - citeste, eventual, starea "detectie purtatoare" pe la modem
 - detecteaza apelul telefonic

Modulul functioneaza incorporat oricarui microsistem unde exista posibilitatea accesarii magistralei de date si de adresa al microprocesorului. O aplicatie optima consta in asigurarea secventierelor necesare de catre un microprocesor (de exemplu MMN 80 - CPU) care contorizeaza intrarup-tile generate de un circuit de temporizare-numarare (eventual de tipul MMN 80 - CTC). Pot fi imaginat si alte posibilitati de configurare in cadrul microsistemului.

Numarul ce urmeaza a fi "format" este stabilit prin program. Resetarea controlerului se face soft sau hard.



Principalele blocuri functionale sint prezentate in continuare.

1. Bloc formare numar - este realizat cu circuitul specializat MN 768 si realizeaza formarea numarului dorit.
2. Bloc comutare - executa inchiderea contactelor necesare pentru formarea unui anumit numar; este realizat cu multiplaxoare analogice integrate.
3. Latch cuvint comanda - retine cuvintul de comanda trimis prin program. Structura acestui cuvint daie o secventa ce urmeaza a fi executata.
4. Blocuri de comanda - realizeaza comanda efectiva a secventei impuse (semnalizare, decuplare telefon, introduce incedanta mica pe linie, cuplare modem); folosind cu precadere relee.
5. Blocuri de adaptare pentru citirea starilor - permit citirea starilor (detectia existentei buclei de curent a centralei, detectie raspuns abonat, detectie apel telefonic etc.).
6. Buffer 3-state - este necesar pentru citirea starilor pe magistrala de date.
7. Decodorul de adrese porturi - decodifica starea magistralei de adrese permitind adresarea porturilor de pe modul.

3. Functionare

Partea cea mai importanta a acestui modul o constituie circuitul specializat MN 768 (Loop Disconnect Dialler) care asigura intreruperea buclei de curent continuu a centralei de un numar de ori egal cu cifra din numarul abonatului chemat care este "formata" la un moment dat. Circuitul asigurand si pauzele corespunzatoare intre cifre. Comanda trimiterii unei anume cifre se face prin inchiderea contactului corespunzator intr-o matrice de 4 linii si 3 coloane (figura de mai jos).

	C1	C2	C3
R1	11	21	31
	1	1	1
	1	1	1
R2	41	51	61
	1	1	1
	1	1	1
R3	71	81	91
	1	1	1
	1	1	1
R4	1	81	1
	1	1	1

Testarea unei anume cifre se realizeaza fizic prin stabilirea unui contact electric intre linia si coloana corespunzatoare, conform reprezentarii alaturate.

Solutia adoptata in cazul dispozitivului prezentat foloseste doua multiplexoare analogice de tipul "4 cu 1" (2 x 1/2 MMC-4851) cu intrarile/iesirile comune legate impreuna. Adresarea multiplexoarelor se face dintr-un latch incarcat de pe magistrala de date a sistemului cu un cuvint de control adecvat. Latch-ul este adresat ca port, la o adresa stabilita prin

jumpare, oriunde în spațiul de adresare al porturilor de către microprocesor. Comenzile necesare sînt prescrise prin înscrisura, în mod similar ca mai sus, a unui cuvînt de comandă pe 4 biți, activarea oricărui bit determinînd lansarea uneia din comenzile asociate, și anume:

1. Decuplarea aparatului telefonic local de la linia telefonică
2. Conectarea unei impedanțe reduse (comparabilă cu impedanța castii telefonice) pe linie pentru ca centrala să "sîntă" starea "receptor ridicat" necesară pentru trimiterea semnalului de linie liberă, (ton).
3. Cuplarea unui codex la linia telefonică pentru a permite comunicatia serială de date.
4. Dezinhibarea multiplexoarelor pentru stabilirea contactului electric asociat unei anumite cifre, conform cuvîntului de control înscris în latch-ul respectiv.

Activarea și dezactivarea comenzilor, precum și temporizările necesare (durate de activare, pauze între comenzi) sînt controlate de microprocesor, fără ca aceasta să înseamnă ocuparea timpului sau de lucru, el putînd să contorizeze, de exemplu, întreruperile externe generate de un circuit numărător-temporizator.

O facilitate importantă a acestui moduli este legată de posibilitatea citirii diferitelor stări:

1. Detectia de linie telefonică activă (detectia semnalului "ton") de informații asupra integrității liniei și permite determinarea momentului cînd poate începe "forarea" numărului.

2. Detectia răspuns abonat cheast (facilitate posibilă în cazul centralilor telefonice automate) - sesizează momentul cînd la abonatul cheast este introdusă o impedanță mică pe linie, adică sesizează apariția stării "receptor ridicat" la abonat.

Cele două situații de mai sus sînt determinate pe principiul cunoașterii polarității curenților în linie prin folosirea a două detectoare conectate antiparalel.

3. Detectia apelului telefonic este posibilă prin recunoașterea semnalului asociat apelului (tranziții de impulsuri cu frecvență de 25 Hz) și setarea unui bistabil a cărui ieșire poate fi citită de microprocesor.

4. Detectia purtătoare (sau citirea oricărui alt semnal, opțional) permite sesizarea existenței în linie a purtătoarei generate de un codex cuplat la linie la postul cheast.

Citirea acestor stări se face periodic, sau de cîte ori este cazul, pe magistrala de date, printr-un buffer 3-stare.

Porturile asociate latch-urilor de comandă și control și buffer-ului 3-stare de citire a stărilor sînt adresate printr-un bloc de decodare adrese, realizat cu o cascada de circuite decodificatoare BCD/zecimal.

Există și o posibilitate de semnalizare optică sau acustică utilă pentru cazul avariiilor, al imposibilității stabilirii legăturii, sau pentru avertizare asupra prelucrării controlului liniei telefonice și, implicit, al blocării unui eventual aparat telefonic.

Dispozitivul poate fi adus oricînd în starea inițială prin reset extern sau prin adresare la portul de resetare.

Toate intrările și ieșirile modului sunt separate galvanic de linia telefonică prin relee și optocuplatoare. Pentru realizare s-au folosit circuite integrate MOS (familia MMC 4000) care asigură un consum redus și o mare imunitate la zgomot.

Echipamentul a fost conceput inițial ca o extindere a posibilităților oferite de microcalculatorul MIND (microcalculator modular cu destinație specială și industrială, produs "Microelectronica"). O utilizare eficientă necesită configuratia minimală alcătuită din modulele MIND UC2 și MIND SIO-CTC. Concepția modulară și conectica standard permit utilizarea facilă în orice microsistem unde există acces la magistralele de sistem.

4. Concluzii

Echipamentul prezentat realizează monitorizarea liniei telefonice, substituind funcțiile unui aparat telefonic și permițând controlul cu ajutorul calculatorului. Avantajul său esențial constă în faptul că permite stabilirea de legături temporare pentru transmisiuni de date pe linii telefonice comutate (prin centrale telefonice) fără a fi necesară concesionarea de linii, folosind posibilitățile sistemului național de comunicație.

Rezumat

Controlerul de linie telefonică este destinat stabilirii de legături temporare pentru transmisiuni de date pe linii telefonice comutate (prin centrale telefonice) fără a fi necesară concesionarea de linii, folosind posibilitățile sistemului național de comunicație. Dispozitivul se caracterizează printr-un consum redus, imunitate ridicată la zgomot, fiabilitate sporită (conferite de folosirea circuitelor MOS), separare galvanică a intrărilor și ieșirilor, grad înalt de integrare și miniaturizare. El poate fi atașat oricărui sistem unde există posibilitatea accesării magistrelor de sistem.

Bibliografie

- Postelnicu P. - Linii și sisteme de transmisiuni telefonice
Ed. tehnică, 1984
- Holsworth B. - Microprocessor Engineering
Ed. Butterworths, England, 1987
- Blakules Th. - Proiectarea cu circuite MSI și LSI standard
Ed. tehnică, 1988
- Ardelean I. - Circuite integrate CMOS, manual de utilizare
s.e., Ed. tehnică, 1986
- Data Book, "Microelectronics", 1987

Echipament pentru transmisiuni de date

Autorii: conf.dr.ing. Braur Adrian
as.ing. Pătorac Dan-Alin
as.ing. Morosan Tudor

Universitatea "Stefan cel Mare" Suceava

1. Introducere

În proiectarea acestui echipament s-a pornit de la necesitatea practică de a facilita comunicatia între sisteme de calcul în aplicații de teleconducere, telegestiune, achiziții de date, comenzi la distanță, transmisiuni teletext, accesare facilă a bancilor de date, etc., folosind posibilitățile oferite de sistemul național de comunicații la ora actuală. Concret, acest echipament permite transmiterea de date semiduplex (half-duplex) pe linii telefonice comutate sau închiriate. Stabilirea legăturii între cei doi abonati telefonici este realizată cu ajutorul unui controler de linie telefonică care nu face obiectul acestei prezentări.

Echipamentul asigură codarea în frecvență a semnalului ce urmează a fi transmis, adaptându-l la banda de frecvență a liniei telefonice. La recepție se realizează decodarea FSK corespunzătoare.

O posibilitate de utilizare este cea din figura de mai jos.



2. Descriere generală

Echipamentul prezentat este un sistem care realizează transmisia serială, half-duplex, pe linie telefonică, la emisie realizându-se o codare FSK (modulare). Partea de recepție asigurând decodarea corespunzătoare (demodulare). Modulul poate fi folosit independent sau atașat oricărui microsistem care posedă interfața serială standard RS 232, fără a necesita acces

la magistralele de sistem.

Structura de baza contine:

- lantul de emisie
- lantul de receptie
- blocuri auxiliare

Blocurile functionale pot fi urmarite pe schema bloc.

Blocurile functionale

a. Structura lantului de emisie

a.1 Oscilatorul - este o structura clasica de oscilator cu cuart care genereaza tactul necesar obtinerii purtatoarei.

a.2 Divizorul comandat - realizeaza divizarea necesara obtinerii frecventelor asociate semnalelor logice "0" respectiv "1". Comanda divizorului se face cu semnalul ce urmeaza a fi transmis.

a.3 Numaratorul divizor - genereaza adresele necesare blocului de memorie.

a.4 Memoria ROM - contine, in binar, valoarea esentiala pentru generarea functiei sinus.

a.5 Convertorul digital-analogic - transforma valoarea binara esentiala din memorie in nivele de tensiune continua, obtinandu-se o forma de unda sinusoidală, cu o rezolutie de 1/256, numarul de pasi fiind prescris din blocul de numarare-divizare, in functie de frecventa oscilatorului cu cuart si de precizia dorita.

a.6 Filtrul trece-jos - realizeaza "netezirea" semnalului obtinut la iesirea convertorului digital-analogic pentru atacarea liniei cu o forma de unda cit mai apropiata de cea ideala.

a.7 Separatorul de sens - este un bloc analogic care permite separarea sensurilor de emisie respectiv receptie precum si adaptarea de impedanta spre linia de transmisie.

b. Structura lantului de receptie

b.1 Blocul pentru reglarea automata a amplificarii - este o structura analogica care mentine constant nivelul semnalului la iesire compensand atenuarea liniei realizind si-o prima filtrare trece banda.

b.2 Adaptorul - formator - a fost introdus pentru a realiza translatiile de nivel necesare precum si transformarea semnalului sinusoidal in semnal dreptunghiular.

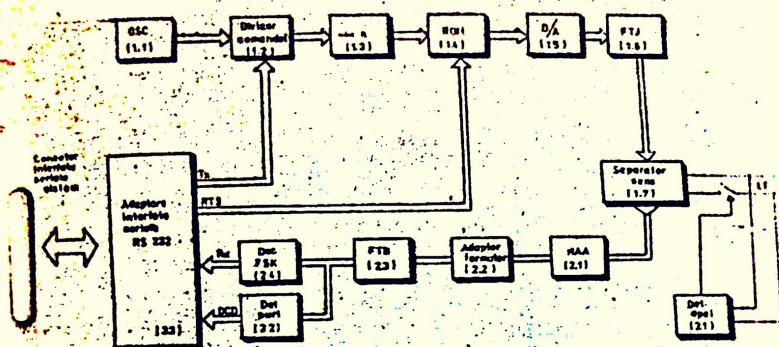
b.3 Filtrul trece-banda - filtru digital ce rejecteaza semnalele cu frecventa in afara benzii utile.

b.4 Decodorul FSK - contine in esenta o bucla PLL cu un indicator de calare, care realizeaza decodarea propriu-zisa.

c. Blocurile auxiliare

c.1 Detectorul de apel telefonic - are o structura care ii permite detectarea apelului telefonic si semnalizarea corespunzatoare prin aprinderea unui LED si, eventual, actionarea unui releu care sa cupleze modemul la linia telefonica.

c.2 Detectorul de purtatoare - sesizeaza existenta purtatoarei si semnalizeaza aceasta prin activarea unui semnal de control si aprinderea unui LED.



EQUIPAMENT PENTRU TRANSMISIUNI DE DATE

Fig. 1

6.3 - Bloc adaptare interfață serială RS 232 - realizează adaptarea conform standardului RS 232 pe partea de comunicație serială.

3. Funcționare

Pentru transmițerea datelor pe linia de comunicație este necesară codificarea informației, codificare care se face prin modulare în frecvență (FSK - frequency shift keying). Conform standardelor internaționale se asociază bitului "0" logic frecvența de 2100 Hz iar bitului "1" logic frecvența de 1300 Hz. Obținerea acestor frecvențe se face prin divizori succesivi, pornind de la frecvența oscilatorului cu cristal de cuarț.

Notăm cu:

- FS - frecvența oscilatorului cu cristal de cuarț
- n8 - factorul de divizare al divizorului comandat, asociat bitului "0"
- n1 - factorul de divizare al divizorului comandat, asociat bitului "1"
- M - factor de divizare al divizorului programabil
- f8 - frecvența la ieșire (emisie) asociată bitului "0"
- f1 - frecvența la ieșire (emisie) asociată bitului "1"

Urmarind structura lantului de emisie si avind in vedere cascadaarea circuitelor de divizare, frecventele de iesire au valorile :

$$f_0 = F_0 / (n_0 \times N)$$

$$f_1 = F_0 / (n_1 \times N)$$

Valorile f_0 si f_1 sint standardizate. Factorul de divizare al divizorului programabil, N , trebuie sa fie egal cu numarul de esantioane inregistrat in memoria ROM. Divizorul programabil furnizeaza, pe maximum 11 biti de adresa (disponibili la o memorie 2716 de 2 ko), in binar, valori numerice crescatoare de la 0 la N . La adresele respective (succesive) in memorie sint inregistrate, in binar, pe 8 biti, valorile esantioanelor pentru generarea functiei sinus. Aceste secvente binare sint convertite intr-o tensiune analogica de un convertor digital-analogic de 8 biti realizat cu circuitul integrat DAC 08. Valoarea unui esantion este :

$$e(p) = k \times (2 \exp 8) / 2 \times [1 + \sin(2\pi p / N) \times p]$$

unde $0 < p < N$, n = nr. natural

k = constanta de proportionalitate dependenta de domeniul maxim de variatie a tensiunii la iesirea convertorului digital-analogic

S-a optat pentru urmatoarele valori :

$$F_0 = 7,2072 \text{ MHz}$$

$$n_0 = 13$$

$$N = 264$$

$$n_1 = 21$$

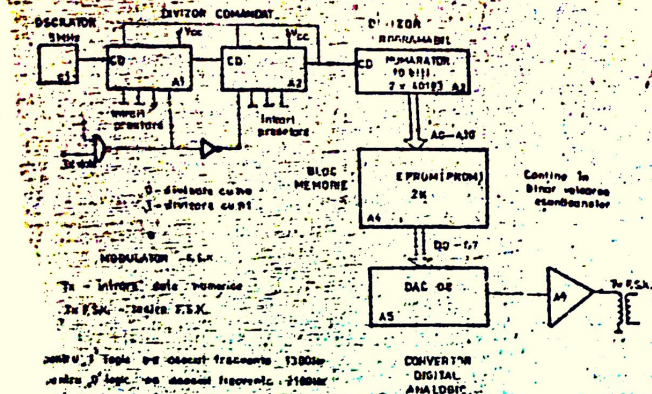
Cu aceste valori, conform relatiilor anterioare, se obtin frecventele dorite $f_0 = 2100 \text{ Hz}$, respectiv $f_1 = 1300 \text{ Hz}$.

Separarea sensurilor de emisie si receptie se face intr-un bloc analogic realizat cu amplificatoare operationale. Separarea galvanica si adaptarea de impedanta spre linia de transmisie este asigurata de un transformator.

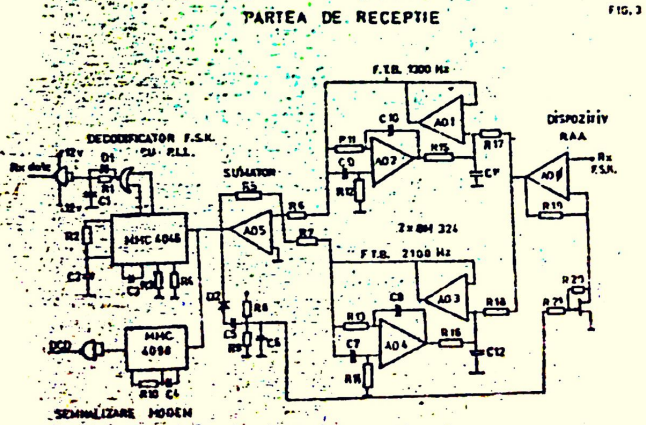
La receptie trebuie sa se discearna intre cele doua frecvente f_0 si f_1 . Pentru aceasta semnalul sosit pe calea de comunicatie este trecut printr-un bloc de filtrare compus din doua filtre trece-banda legate in paralel, avind frecventele centrale f_0 , respectiv f_1 si o latime de banda suficient de mica pentru a minimiza influenta semnalelor parazite. Semnalul astfel prelucrat este aplicat la intrarea unui circuit de tip PLL (phase locked loop - bucla cu care pe faza) avind banda de calare centrata pe frecventa $f_0 = 2100 \text{ Hz}$. Cu un circuit indicator de calare se poate sesiza existenta in linie a frecventei f_0 . Lipsa acestui semnal este interpretata prin inexistenta in linie a unui semnal util, sau prin existenta in linie a frecventei f_1 daca a fost activat si semnalul DCD (Data Carrier Detect - detectie de purtatoare). Semnalul DCD este generat de un monostabil retriggerat continuu de unul din semnalele f_0 sau f_1 .

Pentru asigurarea unui nivel constant la intrarea in circuitul PLL a fost prevazuta o reactia negativa pe amplificatorul de intrare, de tip RAA (reglaj automat al amplificarii).

Echipamentul prezentat este realizat cu componente indigene. S-au folosit circuite integrate MOS din familia MMC 4000 (numaratoare presetabile 40193, circuit PLL 4046, etc.), memorie ROM tip MMN 2716 de 2 ko, convertor digital-analogic DAC 08, partea analogica fiind realizata cu AO tip BM 324.



PARTEA DE EMISIE



4. Concluzii

Gradul ridicat de integrare, minimizarea partii analogice în favoarea celei numerice, utilizarea circuitelor MOS, precum și soluțiile tehnice adoptate atribuie acestui echipament caracteristici specifice: imunitate ridicată la zgomot și perturbatii, consum redus, fiabilitate sporită, depanare ușoară cu posibilitate de auto test, pret scăzut.

Acest model a fost conceput inițial ca o extindere a familiei MIND (microcalculator modular cu destinație specială și industrială, produs "Microelectronica"). Structura modulară și conectica standard permit însă conectarea facilă la orice sistem.

Rezumat

Echipamentul pentru transmisiuni de date este un model half-duplex care permite transmiterea serială a datelor pe linii telefonice. Codarea FSK se realizează prin generarea purtătoare cu un secventiator cu memorie PROM, secventiator atactat cu un semnal de ceas cu frecvență dependentă de valoarea logică a bitului ce se transmite la un moment dat. La recepție, decodarea se face printr-un indicator de calare atactat unui circuit PLL. Soluția tehnică folosită a permis minimizarea volumului de componente, corelat cu un consum scăzut și o mare imunitate la zgomot.

Bibliografie

- Millan I. - Microelectronics
- Grabel R. - Ed. McGraw Hill, S.U.A., 1987
- Stefan Gh.M. - Circuite integrate digitale
- S.A. - E.d.P., 1983
- Blakeslee Th. - Proiectarea cu circuite MSI și LSI standard
- Ed. tehnica, 1988
- Holdsworth B. - Microprocessor Engineering
- Ed. Butterworths, England, 1987
- Ardelean I. - Circuite integrate CMOS, manual de utilizare
- S.A. - Ed. tehnica, 1986
- Vatasescu A. - Circuite integrate liniare, manual de
- utilizare, vol.1, Ed. tehnica, 1979
- S.A.
- Rapmanu R. - Circuite integrate analogice, catalog
- Ed. tehnica, 1983
- Data Book, "Microelectronica", 1989

Interfață specializată pentru achiziții de marii analogice

Autorii: conf.dr.ins. Braur Adrian
as.ins. Potoreac Dan-Alin
as.ins. Morosan Tudor

Universitatea "Stefan cel Mare" Suceava

1. Introducere

Interfața specializată pentru achiziții de marii analogice este un echipament care realizează selectarea sursei de semnal și conversia digitală a marii respective, fiind controlată de un microprocesor de 8 biți. Ea poate fi atașat oricărui microsistem care are magistralele accesibile. Aplicația fundamentală pentru care a fost conceput acest echipament se încadrează în categoria aparatului medical și este un aparat pentru termografie de contact.

Aparatul pentru termografie de contact pune la dispoziție unui sistem de calcul cu posibilități grafice color (din familia PC) datele necesare obținerii unei hărți termice a organismului uman. Harta ce permite depistarea timpurie a diferitelor afecțiuni.

În cazul utilizării unui sistem MIND (microcalculator modular cu destinație specială și industrială produs de "Microelectronica" București) structura minimală cuprinde modulele: UC2 (unitate centrală), SID-CTC (transmisie serială) și cele două module specializate care fac obiectul acestei prezentări, AD12 și MX.

Modulul AD12 - convertor analogic-digital pe 12 biți. Realizează conversia propriu-zisă sub controlul unui soft adecvat.

Modulul MX - multiplexor analogic pentru 64 (32) canale de intrare. Permite selectarea canalului ce urmează a fi "citit".

2. Modulul de conversie analog-digitala pe 12 biti

2.1. Descriere generala

Modulul AD12 este un convertor analogic-digital pe 12 biti cu reactie functionind pe principiul compararii tensiunii de convertit cu o tensiune generata de un convertor D/A de 12 biti comandat cu un latch prescalat prin program. In functie de soft-ul asociat conversia se poate face dupa metoda "cu incrementare (decrementare)" sau "cu aproximatii succesive". Structura hard a acestui modul este organizata in jurul unui circuit PIO care permite controlul resurselor hardware de catre microprocesor.

Blocurile functionale si functiunile lor sint prezentate in continuare.

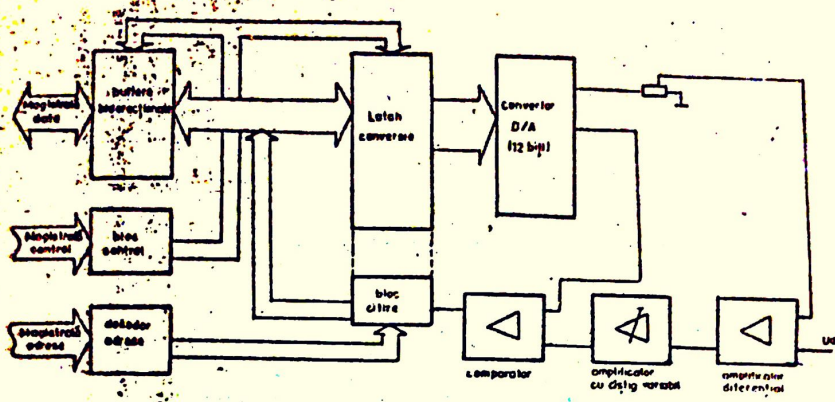
- * latch conversie - retine 12 biti trimisi prin program, necesari blocului de conversie D/A
- * bloc conversie D/A - este realizat cu un convertor specializat pe 12 biti care genereaza tensiunea analogica corespunzatoare, tensiune aplicata blocului de comparare.
- * amplificator diferential de intrare - amplifica $\times 1$ diferenta dintre tensiunea de referinta si tensiunea de convertit.
- * amplificator cu cistig variabil - amplifica $\times 16$ tensiunea diferenta generata de blocul anterior.
- * comparator - sesizeaza printr-un nivel logic corespunzator relatia de ordine dintre valoarea corespunzatoare tensiunii de intrare si tensiunea prescrisa prin program.
- * blocul de citire - permite citirea starii comparatorului (3 - state).
- * buffer-e bidirectionale - asigura bidirectionalitatea si transferile de nivel necesare pentru accesul pe magistrala de date.
- * bloc control - realizeaza adaptarea de nivel pentru semnele de control.
- * decodor adrese - decodifica starea magistralei de adrese permitind adresarea in spatiul porturilor pentru cicustele tip latch si blocul de citire (PIO).

2.2. Functionarea convertorului

Structura hard a modulului de conversie are la baza un circuit MMN 80 - PIO care contine doua porturi de 8 biti. Portul A si 1/2 din portul B asigura 12 biti la intrarea circuitului specializat de conversie digital - analogica pe 12 biti (AD DAC - 80). Ceilalti 4 biti sint folositi conform schemei de mai jos:

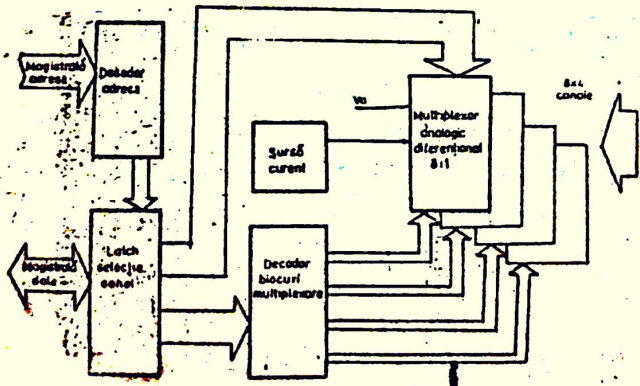
D7 A - D0 A (PIO) = D11 - D4 (AD DAC - 80)
 D7 B - D4 B (PIO) = D3 - D0 (AD DAC - 80)
 D3 B (PIO) = neutilizat
 D2 B (PIO) = neutilizat
 D1 B (PIO) = neutilizat
 D0 B (PIO) = intrare pentru citirea starii comparatorului

15



MODUL DE CONVERSIIE ANALOG- DIGITALA
PE 12 BITI
SCHEMA BLOC

MODUL MULTIPLEXARE
Schema bloc



D3

Pentru conversiile relativ rapide este indicat a se utiliza principiul aproximatiilor successive.

Pentru conversia în zecimal a rezultatului hexazecimal obținut se va avea în vedere capacul de scala pentru AD DAC 88 (5 V). Aspectul de ieșirea convertorului este invers proporțional cu codul hexazecimal aplicat la intrare și numărul de biți pe care se obține rezultatul (eventual în urma unor calcule). Se poate folosi formula:

$$N_{zec} = N_{hex} \times c / (2^{expb} - 1)$$

- b = număr de biți pe care se obține rezultatul (12 de exemplu).
c = capac de scala (5 V).

Factorul de amplificare al amplificatorului cu câștig variabil a fost ales 16 pentru a facilita eventualele calcule în binar (deplasările stînga, dreapta sînt echivalente cu înmulțirea, respectiv împărțirea cu puteri ale lui 2).

Possibilitatea AD DAC 88 de a lucra în cod zecimal nu oferă facilități suplimentare și reduce rezoluția de conversie.

Este posibilă utilizarea lucrului în întreruperi cu $\mu P D$ pentru sesizarea schimbării stării comparatorului, de exemplu.

Reglajele necesare sînt: offset convertor D/A, liniaritate convertor D/A, amplificare semnal analogic și tensiune referință. Pasul conversiei este de $5V / 2^{exp12} = 1,22 \text{ mV}$.

Pentru o amplificare egală cu 16 modulul asigură o rezoluție în tensiune de $76,2 \text{ uV}$ și o rezoluție în temperatură de $7,62 \text{ miligrade Celsius}$.

3. Modulul multiplexor

3.1. Descriere generală

Modulul de multiplexare permite selectarea unui anumit canal analogic dintr-un număr de 32 sau 64, canal pe care se trimite un curent constant, și se citește o tensiune pe o linie analogică comună.

Schema bloc, relativ simplă, este prezentată în continuare:

- * blocul multiplexoarelor - constituie elementul principal al acestui modul și este alcătuit din 4 sau 8 multiplexoare analogice diferențiale cu câte 8 canale, de tipul MMC 4097.

- * latch selecție canal - reține cuvîntul de comandă trimis prin program, cuvînt care, decodificat, determină selecția unui anumit canal. Este realizat cu MMC 4508.

- * decodor blocuri multiplexare - decodifica parțial cuvîntul de comandă și dezinhiba chip-ul multiplexorului corespunzător.

- * decodor adrese - decodifica starea magistralei de adrese asigurînd adresarea în spațiul porturilor. Decodările folosite sînt MMC 4028.

- * sursă curent - asigură curentul constant necesar a fi

trimis canalelor selectate .

3.2. Functionarea multiplexorului

Cuvintul de comanda trimis prin program la adresa care determina selectarea si inscrierea latch-ului (adresa stabilita prin jumper) are structura de mai jos :

D7 - "activ sus , determina comutarea sursei de curent
D6 - "activ sus , valideaza decodificatorul blocurilor
de multiplexare

D5 - D3 = dezinhiba chir - ul multiplexorului ce urmeaza
a fi selectat

D2 - D0 = selecteaza canalul de pe fiecare multiplexor

Selectarea latch - ului se face ca port , clasic .

Pentru selectarea canalului 0 , de exemplu , structura
cuvintului de comanda trimis este :

D7 D6 D5 D4 D3 D2 D1 D0
1 1 0 0 0 0 0 0 00h

Sursa de curent este realizata cu doua tranzistoare in
montaj " oglinda de curent " . In timpul cit nu este folosita
sursa de curent este comutata pe o rezistenta care simuleaza
impedanta sarcinii pentru a reduce oscilatiile de comutare .

Traductoarele folosite in aplicatia "aparatură pentru
termografie de contact" sint 8M 135 (LM 135) (senzori de
precizie pentru determinarea temperaturii in gama -55 , +150
grade Celsius , cu o sensibilitate de 10 mV/grad , liniare in
toata gama) . Polarizarea traductoarelor se face la curent
constant (intre 400 uA si 5 mA) iar caderea de tensiune pe ele
ofera informatii despre marimea temperaturii .

Impedanta dinamica scazuta (1 ohm) determina o mare
inertanta la zgomot chiar in conditiile conectarii traductoarelor
cu conductoare relativ lungi (1 - 3 m) .

Soft-ul adoptat asigura functiile de monitor , selecteaza
periodic traductorul ce trebuie citit , face calibrarea
traductoarelor , realizeaza conversia si executa unele calcule
(medieri , transformarea unitatilor de masura , etc .) . Datele
obtinute sint preluate de un calculator din familia PC-urilor
care face prelucrările ulterioare dorite (deseneaza "harta
termica" propriu-zisa , de exemplu) .

Metoda de diagnosticare care sta la baza aparatului pentru
termografie de contact constituie obiectul preocuparii mai multor
colective medicale din tara noastra .

4. Concluzii

Modulele care alcatuiesc interfata specializata pentru
achizitiile de marii analogice , atasate unui microsistem pe 8
bitti , permit obtinerea unei structuri capabile sa "citeasca"

temperatură cu o precizie de 0.0076 grad Celsius de la un număr de 64 (32 de traductoare (expandabil de un număr de ori egal cu numărul sporturilor disponibile în microsistem) . Aceste facilități fac posibilă obținerea unei hărți termice a organismului uman din analiza căreia se pot depista afecțiunile maligne.

Flexibilitatea structurii prezentate face posibilă implementarea în diverse alte aplicații unde este necesară "citirea" unor mărimi analogice relativ lent variabile în timp (temperaturi, presiuni, etc.) cu o precizie foarte ridicată.

Rezumat

Interfața specializată pentru achiziții de mărimi analogice cuprinde un modul de conversie analogic-digitală pe 12 biți și un număr mare de module multiplexoare pentru selectarea canalului analogic. Conversia este controlată soft și se face prin compararea continuă a tensiunii de convertit cu tensiunea generată de un convertor D/A pe 12 biți. Cu ajutorul unui latch și a logicii de decodare asociate se selectează, prin intermediul unui multiplexor analogic, o anumită sursă de semnal. Aplicația de bază este termografia umană de contact, dar interfața poate fi utilizată pentru orice achiziții de semnale analogice, cu adaptări minime.

Bibliografie

- Holdsworth B. - Microprocessor Engineering
Ed. Butterworths, England, 1987
- National Semiconductor
catalog, 1983
- Institute of Microelectronics, catalog
Bulgaria, 1988
- Rapreanu R. - Circuite integrate analogice, catalog
Ed. tehnica, 1983
- Ardelean L. - Circuite integrate CMOS, manual de
utilizare, Ed. tehnica, 1986
- Data Book, "Microelectronics", 1989

**ALGORITM SI CIRCUIT DEDICAT DE ÎMBUNĂTĂȚIRE A EFICIENȚEI
PROCESULUI DE CORECȚIE ÎN TRANSMISIILE SINCRONE
CU PONDERE CONSTANTĂ**

Autor: As.ing. Rezeanu Ștefan-Cristian, I.P.Iagi

1. Generalități

În cadrul Sistemului mondial de pericol și siguranță pe mare - GMDSS (Global Maritime Distress and Safety System), o importantă fracțiune din volumul legăturilor radio este alocată telegrafiei de bandă îngustă - NBDPT (Narrow-Band Direct-Printing Telegraphy), care utilizează moduri de lucru cu detecția și corecția erorilor ([3]). Pentru exemplificări ne vom fixa asupra comunicațiilor radiotelex, care utilizează transmisii sincrone în cod cu 7 elemente de semnal cu pondere constantă - în combinația de semnale unitare raportul dintre cele două stări semnificative fiind întotdeauna 4/3. Tipul de modulație utilizat este FSK, notațiile CCIR fiind:

- pentru polaritatea START: A în cod CCIT⁹ nr. 2 (Baudot) și B (corespunzător frecvenței superioare) în codul cu 7 elemente de semnal cu pondere constantă;
- pentru polaritatea STOP: Z, respectiv Y (asociat frecvenței inferioare).

Astfel, combinația care reprezintă 0 (în mod literă) sau 9 (în mod cifră) va apare AAAZ în cod Baudot și BYYYBB în cod cu 7 elemente de semnal cu pondere constantă (4B/3Y).

Detecția erorilor presupune verificarea ponderii cuvintelor de cod recepționate, în timp ce corecția se efectuează în mod diferit în funcție de modul de operare:

D4

- modul semiduplex ARQ (Automatic Retransmission Request) utilizează în acest scop retransmisia caracterelor la cererea corespondentului, până la recepționarea lor cu pondere corectă (sau întreruperea legăturii datorită condițiilor improprii);
- modul unidirecțional FEC (Forward Error Correction) recurge la emisia cu diversitate în timp, fiecare caracter fiind transmis de două ori (DX - transmisie directă, RX - retransmisie), existînd subclasa FEC-selectiv, în care polaritatea semnalului de date este inversată (ponderea devenind $3B/4Y$); corecția se efectuează comparînd caracterele RX și DX: dacă unul dintre ele are pondere incorectă, celălalt este imprimat; dacă ambele au pondere incorectă, sau corectă dar caracterele sînt diferite, se imprimă "spatiu".

2. Algoritmul propus

Metodele de corecție expuse prezintă dezavantajul că pot mări uneori nejustificat durata legăturii (ARQ) - putîndu-se chiar lua decizia întreruperii acesteia - sau sînt tipărite mesaje cu multe spații (FEC) în condiții de recepție zgomotoasă.

Algoritmul suplimentar de corecție a erorilor de demodulare primară datorate unei recepții zgomotoase, propus în continuare, necesită sesizarea tranzițiilor nedorite (parazite) ce apar în fluxul de date într-o fereastră de timp centrată pe momentul preluării datei; coroborînd aceste informații (în modul de operare FEC) cu datele de pe pozițiile RX și DX eșantionate de tactul refăcut, se poate obține îmbunătățirea eficienței procesului de corecție.

În diagrama de impulsuri ilustrativă din fig. 1 (unde notațiile corespund schemei bloc din fig. 2), s-a presupus că este folosit codul cu 7 elemente de semnal cu pondere constantă (31), emitîndu-se caracterul "0" (sau "9") în mod de lucru FEC-selectiv. Vom presupune, de asemenea, că sincronizatorul de bit a depășit faza de achiziție, tactul refăcut (4 C) fiind corect fazat cu frontul negativ în imediata vecinătate a momentelor în care pot apare tranziții în fluxul de date primar demodulate.

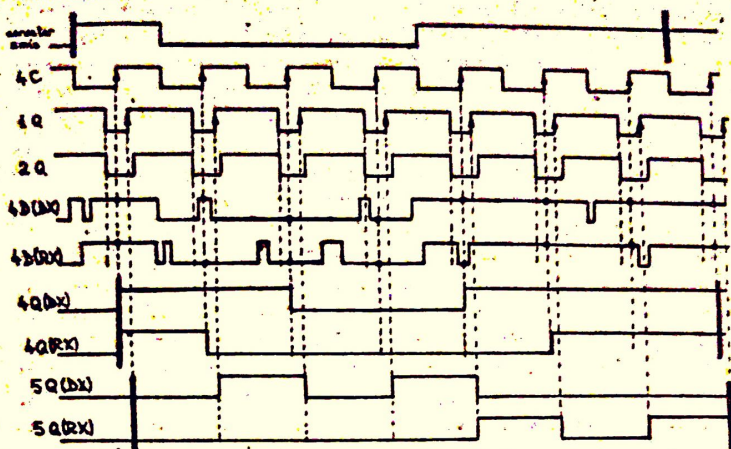


Fig. 1. Forme de undă ilustrative pentru algoritmul propus.

Caracterul este recepționat o dată în transmisie directă (4D-DX) și a doua oară în retransmisie (4D-RX), dar afectat de zgomot, prin scurte impulsuri parazite. Suplimentar fereastră de timp (1 Q) centrată pe momentul preluării datei (moment marcat de frontul pozitiv al tactului refăcut, unde este maximă probabilitatea unei interferențe intersimbol minime), circuitul dedicat - descris la paragraful 3 - necesită și o fereastră de timp asimetrică, cu frontul crescător întârziat (2 Q). Frontul crescător al ferestrei centrate (1 Q) va marca momentul în care se decide dacă pe durata ei au existat sau nu tranziții în fluxul de date primar demodulate. În timp ce frontul crescător al tactului refăcut decide asupra valorii logice a biților succesivi emiși (starea în care găsește acest front datele 4D-DX și 4D-RX este marcată cu punct în fig. 1).

Am presupus că a fost emis caracterul având, în ordine, stările semnificative 1000111 (YBBVYY, deci cu pondere 3B/4Y, corectă în situația lucrului în mod FEC-selectiv).

Se observă că în condiții de zgomot se recepționează 1100111, cu pondere incorrectă 2B/5Y (4Q-DX) și 1000011, cu pondere incorrectă 4B/3Y (4Q-RX). Conform recomandărilor CCIR, se va tipări "spatiu", luându-se decizia de caracter eronat recepționat. În absența informațiilor privind tranzițiile în fereastră (date notate prescurtat DTF), deși se observă imediat că stările semnificative cu numerele de ordine 1, 3, 4, 6 și 7 au fost identic recepționate în cele două cuvinte (1 00 11), din faptul că ponderea trebuie să fie 3B/4Y, nu se poate concluziona decât că s-ar fi putut transmite fie 1100011, fie 1000111, fără posibilitate de decizie imediată (evident, exceptăm posibilitatea de decizie pe baza examinării ulterioare a cuvintelor recepționate). Dacă, însă, se folosesc și cuvintele DTF (5Q-DX: 0101000 și 5Q-RX: 0000101), excluzând tranzițiile 1e pe stările identic recepționate (4 pe DX și 7 pe RX) pe considerentul că există o probabilitate maximă ca tranziția să nu fi afectat valoarea logică corectă în momentul eșantionării, tranzițiile rămase ne vor conduce la ambele cuvinte - prin schimbarea valorii logice a stării 2 pe DX și 5 pe RX - la același caracter, cu pondere corectă 3B/4Y, chiar caracterul emis 1000111.

3. Circuitul utilizat

Circuitul prezentat la nivel de schemă bloc în fig. 2, permite utilizarea metodei suplimentare de corecție descrise în paragraful anterior prin memorarea existenței unor tranziții parazite a datelor în momente apropiate celui în care acestea se prăiau. Suplimentar, prin eventuala contorizare a stărilor semnificative pe durata cărora au apărut tranziții în fereastra generată, circuitul propus poate oferi și informații asupra stadiului procesului de calare a sincronizatorului de bit, permițând controlul vitezei de calare a acestuia (uzual în sensul scăderii ei după efectuarea rapidă a achiziției).

Circuitul folosește ca intrare datele primar demodate, care atacă atât blocul de memorare a datei și cit și detectorul de tranziții al sincronizatorului de bit, presupus a furniza scurte impulsuri pozitive la fiecare tranziție din fluxul de date.

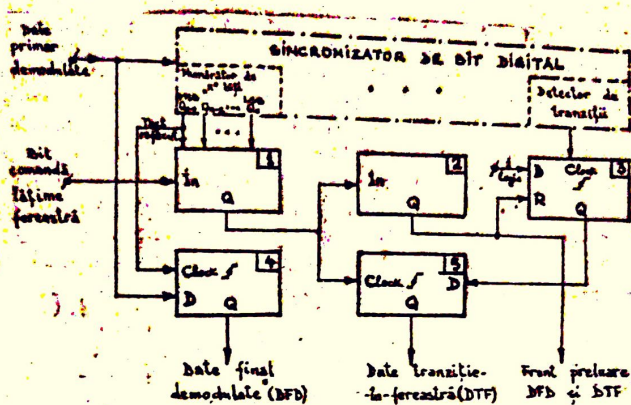


Fig. 2. Schema bloc a circuitului propus.

Numărătorul divizor de frecvență prin "2", furnizează tactul răfăcut (4 C), al cărui front pozitiv eșantionează datele de la intrarea blocului 1. Numărătorul pe "n" biți și detectorul de tranziții sînt uzuale în sincronizatoarele de bit digitale ([1], [2], [4]), numărătorul fiind obligatoriu, dacă topologia de sincronizator de bit nu conține blocul detector de tranziții, acesta va fi realizat separat, fiind obligatoriu în cazul soluției de circuit propuse.

Ieșirile numărătorului atacă generatorul de fereastră de lățime variabilă 1, lățimea ferestrei fiind presupusă cu două valori prestabilite, prescrise prin intermediul bitului de comandă a lățimii ferestrei. În fereastra generată (pe durata stării 0 logic - 1 Q) se va analiza prezența unor tranziții nedorite în fluxul de date primar demodulate.

Blocul de întârziere al frontului pozitiv al ferestrei 2, furnizează blocului detector de tranziții în fereastră 2, semnalul de reset (2 Q), permițînd astfel blocului de "memorare tranziție în fereastră" 3, memorarea - în momentul frontului pozitiv al ferestrei (1 Q) - existenței a cel puțin unei tranziții pe durata ferestrei.

Existența pe durata ferestrei a unui front pozitiv la intrarea blocului detector de tranziții în fereastră (3C-DX pentru transmisia directă și 3C-RX pentru retransmisie), provoacă trecerea nivelului 1 logic de la intrarea de date a blocului 3, la intrarea de date a blocului 5 (5D-DX și respectiv 5D-RX). Caracterele formate din datele final demodulate DFD (4Q-DX și 4Q-RX) și respectiv datele asupra tranziției în fereastră DTF (5Q-DX și 5Q-RX), vor fi preluate în vederea analizei și deciziei la frontul pozitiv întârziat al ferestrei (2 Q).

4. Concluzii

Circuitul propus oferă informații rapide asupra stadiului procesului de calare, permițând, prin generarea ferestrei variabile de analiză a prezentei tranzițiilor și utilizarea algoritmului de corecție descris, demodularea cu randament îmbunătățit a datelor recepționate; se scurtează durata legăturii prin micșorarea numărului cererilor de repetiție (ARQ), respectiv a numărului de spații tipărite în urma deciziei de "caracter eronat recepționat" (FEC).

BIBLIOGRAFIE

1. BRANA CODRUȚ, BRANA VALERIA: "Transmiterea informației numerice - vol. II", Editura Tehnică București 1970, Cap. 9 - "Sincronizarea de bit (sinfazarea)", pag. 79-83;
2. GREEN W. T. Jr, KEAN BILL: "Digital Phase-Locked Loops move into analog territory" - Electronic Design, March 31, 1982, pag. 95-100;
3. "Recommendations et rapports du CCIR, 1986", volume VIII-2, Geneva, 1986, ISBN 92-61-02782-2 - Recommendation 490, pag. 76-77 și Recommendation 625, pag. 5-59;
4. SMITH MARK: "Modern Communication Circuits", Mc Graw-Hill 1986, parag. 8.7 - "Digital Phase-Locked Loops", pag. 324-326.

CIRCUIT DE INTRODUCERE OPTIMĂ A CORECȚIEI DE FAZĂ,
APĂRINT UNUI SINCRONIZATOR DE BIT DIGITAL
CU URMĂRIREA TRANZIȚIILOR

Autor: As.ing. Rezeanu Ștefan-Cristian, I.P.Iași

1. Generalități

Circuitele sincronizatoare de bit pentru comunicații sincrone permit obținerea în receptor a unui semnal de referință în concordanță de fază cu tactul folosit la emisia datelor (valoarea " f_c " a frecvenței tactului fiind cunoscută). Refacerea cu precizie a fazei tactului de emisie, permite eșantionarea fluxului de date primar demodulate la momente de timp care să minimizeze interferența intersimboluri (conform unuiu dintre criteriile lui Nyquist), maximizînd astfel probabilitatea de decizie corectă asupra valorii logice a bitului emis.

Sincronizatoarele de bit cu urmărirea tranzițiilor extrag informația de sincronizare direct din semnalul de date, acesta din urmă trebuind, deci, să prezinte un număr suficient de tranziții. Sincronizatoarele de bit digitale fac aproape în exclusivitate apel la așa-numita metodă de reglaj, care constă în corecția periodică a fazei tactului generat local, prin introducerea unui salt de fază fix la fiecare tranziție a semnalului de date recepționat.

Necesitînd rezolvarea problemei de ajustare a fazei unui semnal generat local, configurația care s-a impus în sincronizarea de bit digitală este cea de DPLL - buclă digitală cu calare de fază ([4]), unul dintre cele mai utilizate circuite integrate de acest tip fiind 54 LS 297 (descriș pe larg în [3]).

DS



În lipsa tranzițiilor din fluxul de date recepționat, numărătorul binar de ieșire - de "n" biți - livrează tactul refăcut (de frecvență " f_n ") cu un coeficient de umplere de 50%. Vom presupune că tranzițiile sînt așteptate la frontul negativ al

tactului refăcut. În această situație frontul pozitiv fiind utilizat la eșantionarea datei recepționate; de asemenea, rezultă că dacă tranziția găsește tactul refăcut în starea logică "high", se impune corecția prin avansul fazei tactului generat, pentru a se realiza apropierea frontului negativ al acestuia de poziția tranziției, presupusă utilă (problema protecției la tranziții parazite va fi discutată ulterior); în mod corespunzător, corecția se efectuează cu întârziere a fazei dacă tranziția găsește tactul refăcut în starea logică "low".

Metoda propusă ar putea fi denumită a ferestrelor de timp sincronizate, necesitând generarea a două ferestre de timp cu duratele (T_1 și T_2) corelate între ele și cu perioadele impulsurilor de tact (de frecvențe " $2^{m+1} \times f_c$ " și " $2^n \times f_c$ "), după relația:

$$\frac{3}{4} \times T < T_1 < T_2 < T = \frac{1}{2^n \times f_c}$$

Justificarea inegalităților de mai sus precum și a afirmației că fereastra de corecție T_1 trebuie sincronizată cu impulsurile de frecvență " $2^{m+1} \times f_c$ " și " $2^n \times f_c$ ", începînd din starea 0 logic a acestora și încheindu-se tot în starea 0 logic a impulsurilor de frecvență " $2^n \times f_c$ " - care sînt utilizate la obținerea tactului refăcut în afara ferestrei T_2 de corecție - se va face apelînd la fig. 2.

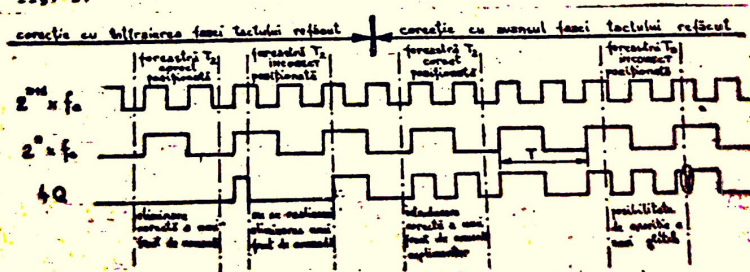


Fig. 2. Poziționarea necesară a ferestrei de corecție T_1 .

Întirzirea fazei tactului refăcut se realizează prin eliminarea în întregime a unui impuls pozitiv la intrarea de clock a numărătorului binar de "n" biți, ceea ce duce la suprimarea unui front pozitiv de comandă, în timp ce avansul fazei se realizează prin introducerea a două impulsuri de frecvență dublă în locul unui singur impuls pozitiv, asigurând existența unui front pozitiv suplimentar de comandă a incrementării valorii binare a ieșirii acestuia numai în cazul poziționării corecte a ferestrei de corecție T_2 . În consecință, rezultă că în ambele cazuri se introduce un salt de fază limitat ca valoare la $(360/2^n)$ grade.

Se poate face observația că, în vederea obținerii unei creșteri a vitezei de calare, fereastra de corecție T_2 poate fi prelungită cu un multiplu al perioadei T , pentru a ne asigura că sfârșitul ei găsește tot în 0 logic impulsurile de frecvență $2^n \times f_c$; această ultimă condiție permite, de exemplu, evitarea apariției unui scurt impuls "low" parazit la sfârșitul ferestrei de corecție T_2 (această posibilitate fiind ilustrată în fig. 2, ultima fereastră de corecție), în cazul comutării (blocul 4, fig. 3) din starea 1 logic a impulsurilor de frecvență $2^{n-1} \times f_c$ tot în starea 1 logic a celor de frecvență $2^n \times f_c$, dacă 3Q ajunge în "low" în avans față de trecerea în "high" a lui 3Q. Oricum, după achiziția inițială viteza de calare trebuie micșorată, saltul de fază fix prescris la fiecare tranziție împiedicînd ieșirea rapidă din sincronism a tactului refăcut la o eventuală tranziție parazită.

Cele două ferestre de timp sînt generate neretriggerabil, din următoarele considerente:

- tranziții prea apropiate în timp (pe care le vom denumi, în continuare, tranziții adiacente), condiție care ne asigură că cel puțin una este parazită, vor genera o singură fereastră T_1 ;
- condițiile impuse asupra duratei ferestrelor ($T_2 - T_1 > (3/4) \times T$), includ posibilitatea apariției a două impulsuri negative (pe durata T_1) la intrarea generatorului ferestrei T_2 , care vor genera, însă, o singură fereastră de corecție T_2 în care se va opera asupra tactului refăcut.

3. Implementarea la nivel de circuit

Soluția propusă permite implementarea într-un circuit de o complexitate redusă, prezentat în fig. 3, care utilizează ca intrări fluxul de date primar demodulat și două tacte, cu coeficient de umplere 50%, de frecvențe $2^{nd} \times f_c$ și $2^a \times f_c$, generate extern. Circuitul de introducere sincronizată a corecției de fază este constituit din blocurile 1-4, blocurile suplimentare figurate (5-7) întregind o configurație de sincronizator de bit realizată în jurul circuitului propus.

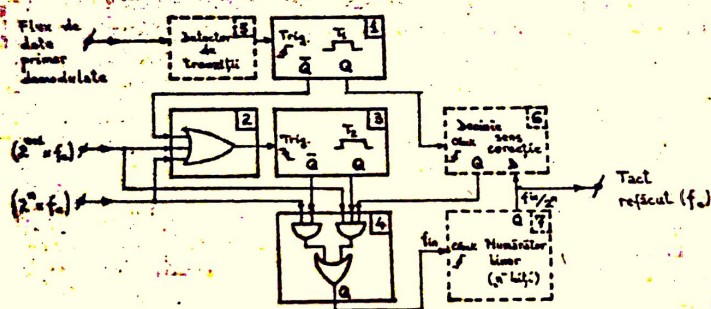


Fig. 3. Schema bloc a circuitului realizat.

Detectorul de tranziții (5), furnizează scurte impulsuri pozitive, cu durata mult mai mică decât T_1 , la fiecare tranziție din fluxul de date recepționate, impulsuri al căror front pozitiv comandă generatorul neretriggerabil al ferestrei de durată T_1 (1), care astfel realizează filtrarea unor eventuale tranziții adiacente care ar apărea pe durată T_1 . Fereastra negativă (1-0), împreună cu impulsurile de frecvență $2^{nd} \times f_c$ și $2^a \times f_c$, generează printr-o poartă SAU (2) impulsuri al căror front negativ declanșează generatorul neretriggerabil al ferestrei sincronizate de corecție T_2 (3). La frontul crescător al ferestrei pozitive T_1 (1-0), circuitul de eșantionare și stabilire a sensului de corecție 6 memorează (6-0) semnalul tactului în care a apărut tranziția.

În lipsa ferestrei de corecție T_c , blocul de selecție și introducere a corecției (4), furnizează numărătorului divizor de frecvență prin 2^n (7), impulsuri de frecvență $2^n \times f_c$, pe durata unei ferestre de corecție T_c , numărătorului îi este furnizat un tact (1-0) ce depinde de starea logică a lui 6-0: impulsuri de frecvență $2^n \times f_c$ pentru 1 logic și respectiv tact înghețat în 0 logic pentru 0 logic. Deci, prin modificarea lungimii semialternanțelor tactului refăcut cu cîte o perioadă T corespunzător fiecărei ferestre de corecție, se acționează în sensul apropierii frontului negativ al acestuia de tranzițiile din fluxul de date demodulate, scăderea factorului de divizare al numărătorului ducînd la creșterea vitezei de calcul.

4. Concluzii

Circuitul ce implementează principiul propus prezintă avantajul unei simplități structurale deosebite, permițînd realizarea comodă a unui circuit integrat dedicat. Suplimentar, probabilitatea de ieșire din sincronism - în cazul unei recepții zgometoase - este redusă prin filtrarea preliminară parțială a tranzițiilor adiacente.

BIBLIOGRAFIE.

1. BENNETT R. WILLIAM, DAVEY R. JAMES: "Data Transmission", Mc Gray-Hill 1965, parag. 14-1 - "Symbol timing for a synchronous binary system", pag. 260-262;
2. BRANA CODRUȚ, BRANA VALERIA: "Transmiterea informației numerice - vol. II", Editura Tehnică București 1970, Cap. 9 - "Sincronizarea de bit (sinfazarea)", pag. 79-83;
3. GREEN W. T. Jr, KEAN BILL: "Digital Phase-Locked Loops move into analog territory" - Electronic Design, March 31, 1982, pag. 95-100;
4. SMITH JAMES, "Modern Communication Circuits", McGraw-Hill 1986, parag. 8.7 - "Digital Phase-Locked Loops", pag. 324-326.

METODA NECONVENTIONALĂ DE ACHIZITIONARE A INFORMAȚIEI

DESPRE TEMPERATURA

As.ing. GALUȚA NICOLAE

Catedra de Calculatoare

Facultatea de Automatică și Calculatoare

Institutul Politehnic Iași

Ing. MAYAYA KILOLE - KAKWANA, doctorand

Facultatea de Automatică și Calculatoare

Institutul Politehnic Iași

INTRODUCERE

Lucrarea de față se încadrează într-o lucrare mai vastă axată pe executia de măsuratori într-un mediu industrial. Ceea ce vom prezenta se referă la achiziționarea informației despre temperatura în cazul unui mediu cu temperatura lent variabilă. Elementul de noutate constă în faptul că desi folosim ca traductor un termistor (element cu caracteristica puternic neliniară) nu vom folosi pentru conversia uzuală AD urmînd apoi aplicarea unei tehnici de liniarizare, ci în faptul că prin înșirarea metodei folosite informația rezultantă este caracterizată de liniaritate.

PREZENTAREA METODEI

Vom folosi ca traductor un termistor a cărui caracteristică este descrisă de relația:

$$R(\theta) = A e^{B/\theta}$$

unde A și B sînt două constante de material.

Executînd o conversie în tensiune și o amplificare vom obține o tensiune dependentă de temperatura după relația:

DC

$$U(t) = U_0 e^{-t/R_1 C}$$

Elementele de baza pe care ne vom baza pentru liniarizare vor fi tensiunile de incarcare si descarcare a unui condensator. Principial circuitul de masura este prezentat in fig. 1.

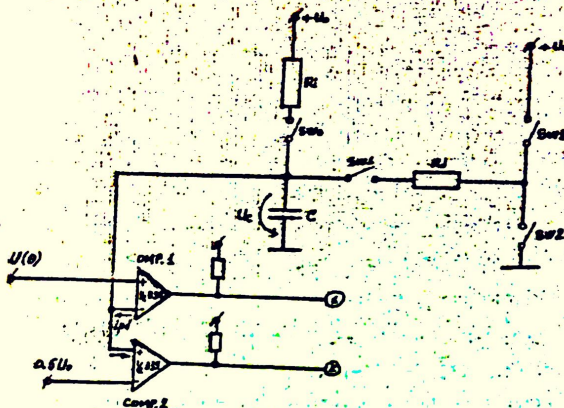


Fig. 1

Facea la inceput mentiunea ca sistemul este conectabil la un sistem cu microprocesor - deci iesirile notate cu 1 si 2 vor fi citite de catre acesta.

Presupunem initial ca $U_c = 0V$ si $S_{W0} = ON$, $S_{W1} = S_{W2} = S_{W3} = OFF$. Condensatorul se incarca pe circuitul $+U_0 - R_1 - C - GND$. Legea de variatie a tensiunii la borne este descrisa de relatia :

$$U_c(t) = U_0(1 - e^{-t/R_1 C})$$

Un parametru caracteristic este produsul $R_1 C$. Il vom determina prin intermediul COMP2. U_c este tot timpul comparata cu $0.5U_0$ -

33
 (pătrăsc temperatură) fiind citite de microprocesor a.i. acesta
 din urme contorizează timpul scurs din momentul $U_c=0V$ pînă la
 momentul $U_c=0.5U_0$. Notăm acest timp cu t_1 .

$$U_c(t_1)=0.5U_0 \Rightarrow 0.5U_0=U_0(1-e^{-t_1/R_1C})$$

$$\Rightarrow \frac{t_1}{R_1C}=2 \Rightarrow t_1/R_1C=\ln(2) \Rightarrow R_1C=t_1/\ln(2)$$

În continuare vom determina printr-o metoda similară produsul
 R_2C . Pentru aceasta vom produce $SW_0=OFF$, $SW_1=SW_2=ON$, $SW_3=OFF$.
 Pentru un timp acoperitor a.i. să putem din nou face presupunerea
 că $U_c=0V$ după care producem $SW_0=SW_2=OFF$, $SW_1=SW_3=ON$ provocînd
 încărcarea condensatorului pe traseul $+U_0-R_d-C-GND$. La fel se
 contorizează timpul în care se atinge $0.5U_0$ și-l notăm cu t_2 .

$$U_c(t_2)=0.5U_0 \Rightarrow 0.5U_0=U_0(1-e^{-t_2/R_dC})$$

$$\Rightarrow \frac{t_2}{R_dC}=2 \Rightarrow t_2/R_dC=\ln(2) \Rightarrow R_dC=t_2/\ln(2)$$

În continuare vom provoca la fel ca mai sus descărcarea
 condensatorului a.i. să putem presupune că $U_c=0V$. Din acest moment
 putem să începem a executa măsurători lucru pe care îl vom face în
 doi pași.

PAS 1 : $SW_0=ON$, $SW_1=SW_2=SW_3=OFF$

Condensatorul începe să se încarce. U_c va atinge nivelul U_0
 după un timp infinit. Pentru a putea face măsurătoarea în timp
 finit, alegem un factor $k < 1$ urmărind încărcarea condensatorului
 pînă la valoarea kU_0 . Timpul necesar încărcării condensatorului
 va fi :

$$kU_0=U_0(1-e^{-t_c/R_1C}) \Rightarrow t_c=R_1C\ln(1/(1-k))$$

$$\Rightarrow t_c=R_1C\ln(1/(1-k))/\ln(2)$$

Deci microprocesorul care controlează interfața va menține

34
configuratie switch-urilor pentru timpul t_c dupa care trecem la pasul 2 al procesului de masurare.

PAS 2 : $BN1=BN2=OFF$, $BNO=BN3=ON$

Condensatorul se va descarca legea de variatie a tensiunii la borne fiind descrisa de relatie :

$$U_c(t) = kOUO e^{-t/RdC}$$

Prin intermediul CMP1 comparas $U_c(t)$ cu $U(\theta_x)$. Cit timp $U_c(t) > U(\theta_x)$ microsystemul contorizeaza timpul. Dupa t_x avem :

$$U_c(t_x) = U(\theta_x) \Rightarrow kOUO e^{-t_x/RdC} = D \cdot B/\theta_x \Rightarrow$$

$$\Rightarrow e^{(t_x/RdC + B/\theta_x)} = kOUO/D \Rightarrow$$

$$\Rightarrow \theta_x = BRdC / (RdC \ln(kOUO/D) - t_x) \Rightarrow$$

$$\Rightarrow \theta_x = (Bt2/\ln(2)) / ((Bt2/\ln(2)) \ln(kOUO/D) - t_x) \Rightarrow$$

$$\Rightarrow \theta_x = Bt2 / (Bt2 \ln(kOUO/D) - t_x \ln(2))$$

Presupunind ca microsystemul contorizeaza timpul dupa un ceas pilot de frecventa ν deci perioada τ vom putea exprima t_1, t_2, t_x functie de perioada τ si trei numere $N1, N2, Nx$ dupa cum urmeaza :

$$t_1 = N1\tau, \quad t_2 = N2\tau, \quad t_x = Nx\tau$$

$$\theta_x = BN2\tau / (BN2\tau \ln(kOUO/D) - Nx\tau \ln(2))$$

Notind $B \ln(kOUO/D) = \alpha = \text{const.}$ si $\ln(2) = \beta = \text{const.}$ avem :

$$\theta_x = BN2 / (\alpha N2 - \beta Nx)$$

unde $N2$ si Nx su fost anterior determinate cu ajutorul microsystemului.

Sa vedea care ar fi eroarea introdusa de comparatoare. Curentii lor de polarizare nu influenteaza curentii de incarcare

si descarcare al condensatorului. Presupunem o eroare de comparare in tensiune de valoarea ΔU .

$$U(t) = \Delta U + U_c(t_x) \quad *$$

$$D = B/B = \Delta U + kOU_0 e^{-t_x/RdC} \quad *$$

$$B/B = \ln(\Delta U/D + (kOU_0/D)) e^{-t_x/RdC} \quad *$$

$$D = B/\ln(\Delta U/D + (kOU_0/D)) e^{-t_x/RdC} \quad *$$

$$\Delta \theta = (B/\ln(\Delta U/D + (kOU_0/D)) e^{-t_x/RdC}) - (BRdC/(\ln(kOU_0/D) - t_x))$$

Studiind monotonia lui $\Delta \theta$ ne rezulta pentru o valoare minima a erorii o valoare maxima pentru k_0 . Tinind cont de faptul ca k_0 intervine si in relatile timpilor de incarcare/descarcare al condensatorului C alegem $k_0=0,9$.

Structura bloc a sistemului de masura este prezentata in fig. 2.

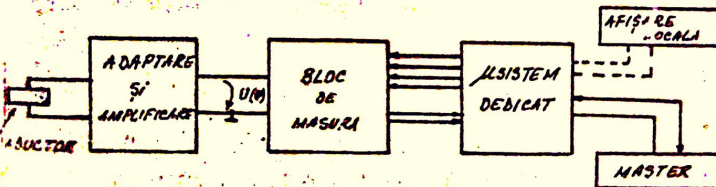
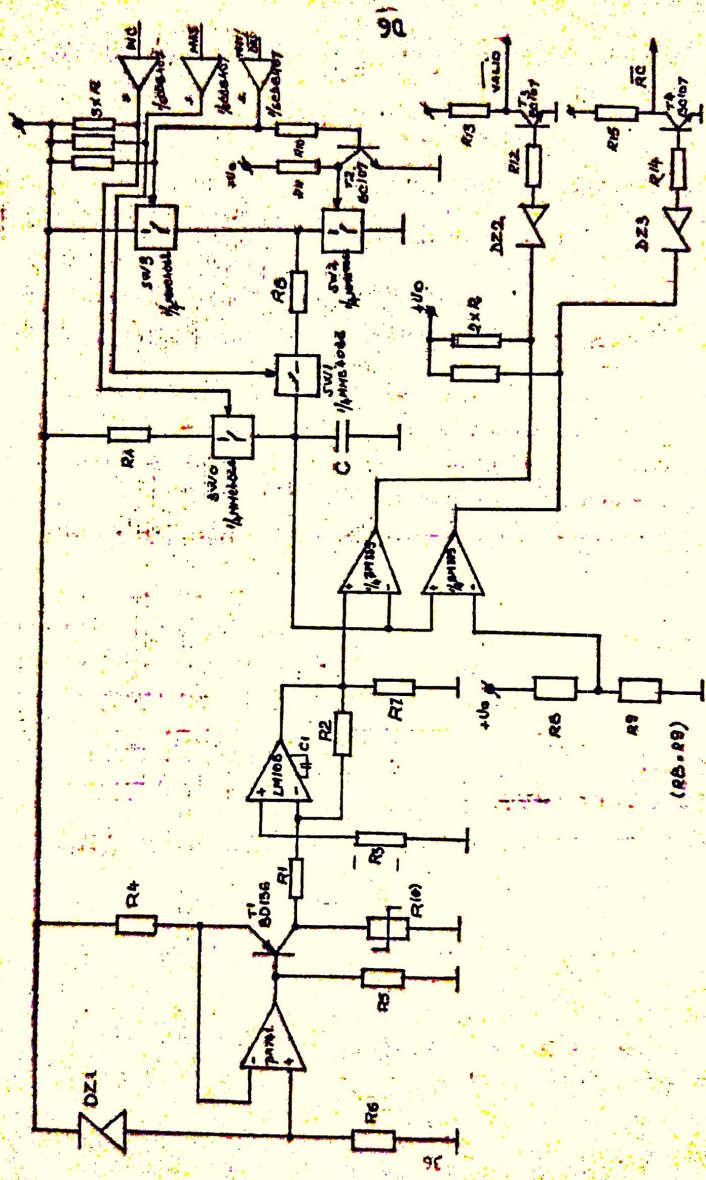


FIG. 2

Structura ce a fost implementata pentru testarea metodei este prezentata in fig. 3. Mentionam ca am folosit un calculator cu

DE



SINTEZA AUTOMATELOR SECVENTIALE UTILIZIND ARBORI MULTIVALENTI

Lălia Feștilă, Sorin Nintea
Institutul Politehnic Cluj-Napoca

REZUMAT

Se prezintă o metodă generalizată de sinteză a circuitelor de tip automat secvențial sincron. Procedența utilizează funcții logice multivalente reprezentate prin arbori de decizie logice. Metoda se poate aplica atât pentru automatele secvențiale binare cât și pentru automatele multivalente implementate cu multiplexoare (parti 1) și bistabile 0 (multistabile 0) și este direct aplicabilă calculului automat.

1. Introducere

În cele ce urmează se prezintă o extindere a procedenței de sinteză a CLC bazat pe logica multivalentă [2] la sinteza automatelor secvențiale sincrone, multivalente, inclusiv a celor binare. Generatorul noii stări are la bază un arbore de decizie fundamental, având o configurație simplă cu două nivele. El poate fi restructurat pe orizontală și pe verticală pentru a fi adaptat la configurația celulelor de tip MX_{rr} [2] impuse. Notatiile utilizate, modificate parțial față de cele uzuale din descrierea diagramei de decizie binară [1] superează o legătură directă față de diagramele de tranziție aferente automatelor secvențiale binare devenind astfel mai intuitive și mai ușor de transpus în logica de calcul automat.

2. Descrierea automatului secvențial în logica multivalentă

În cazul cel mai general automatul sincron este multivalent și are schema bloc dată în figura 1. Semnalul u de ieșire depinde atât de semnalul i de intrare cât și de starea prezentă q . Schimbarea stării are loc în prezența impulsului de tact CK . Între două impulsuri de tact starea este memorată de multistabilul 0.

Proiectarea automatului de acest tip se referă de fapt la determinarea configurației blocului combinational, care stabilește relația logică între semnalul de ieșire și mărimile de intrare și de stare și care determină totodată starea viitoare. Deoarece acest circuit este de tip combinational, pentru sinteza lui se poate aplica un procedeu bazat pe arborii multivalenți de decizie [2]. Rămâne să se indice modul de descriere logică a blocului combinational în aceluiași mod.

Automatele de acest tip sînt în general complexe, cu un mare număr de stări și semnale de intrare și sînt descrise de funcții logice de mari dimensiuni. Cu toate acestea, modelul propus în prezenta lucrare ține o formă primară simplă, bidimensională, datorată introducerii unei relații logice intermediare multivalente.

Plecând la observația că orice numărator de stare conține un număr de m stări distincte, iar tranziția din fiecare stare, prezentă se face spre cel mult n direcții, în funcție de unul sau mai multe semnale de comandă. De exemplu, număratorul de stare din figura 2 are 7 stări distincte ($m=7$) și cel mult 3 direcții d de tranziție ($n=3$). Direcția de tranziție s-a considerat a fi o variabilă n -valentă. Fizic, ea reprezintă un semnal de intrare k -valent ($k \leq n$) sau un vector de semnale în cazul lui [2].

Număratorul de stare poate fi descris de relația implicită:

$$q^* = f(q, d), \quad f: X \times D \rightarrow P \quad (1)$$

unde

$$d = g(q, i), \quad g: Q \rightarrow D \quad (2)$$

$$Q = \{0, 1, \dots, m-1\}, \quad D = \{0, 1, \dots, n-1\}, \quad P = \{0, 1, \dots, p-1\}$$

$$X = \{0, 1, \dots, k-1\}, \quad i = \{i_1, i_2, \dots, i_n\}$$

Relația (1) se consideră să fiind fundamentală și f se poate atasa un arbore cu două nivele. În acest sens există două variante, după cum terminatiile sînt $f(j, d)$; $f(j, i)$ sau $f(q, i)$, $f(j, i)$ unde $j \in Q$, $i \in D$.

Relația (2) se poate reprezenta printr-un arbore cu un nivel, avînd ca terminații semnalele i . Pentru descrierea completă a automatului se atasează și arborele corespunzător funcției de ieșire $U = h(q, i)$. În funcția de timp multiplexarea arborilor se pot restructura, descompunîndu-i pe orizontală și pe verticală

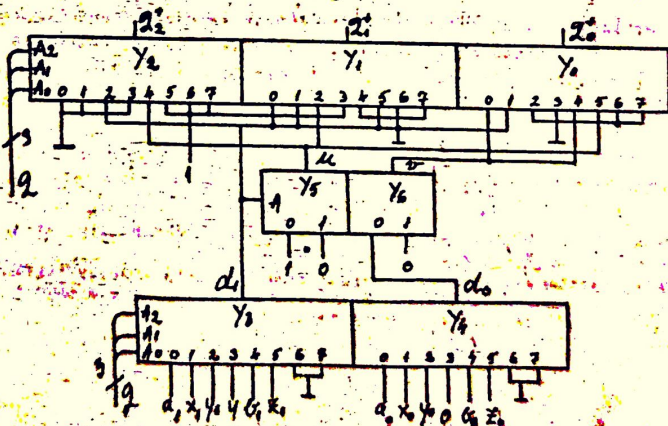
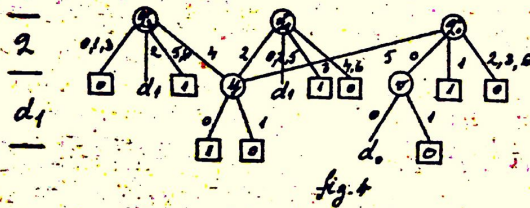
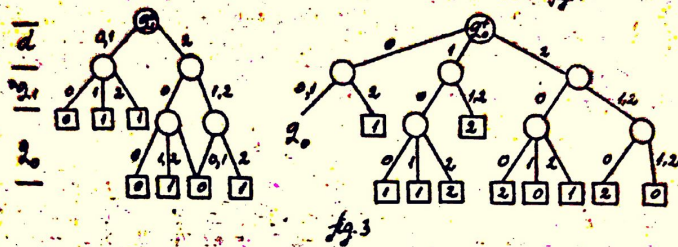
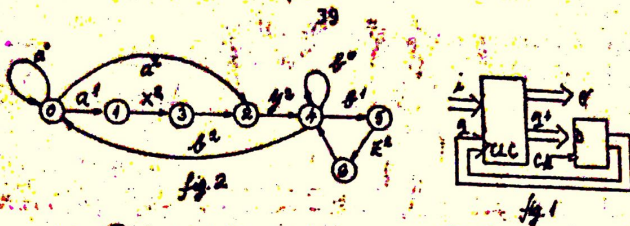


fig. 5

VI

Pentru a fi adus în configurația dorită

3. Exemple

Funcțiile (1) și (2) ale numerarului de stare dat în figura 2 sunt indicate în tabelul 1

$q^* = f(q, d)$								$d = g(q, 1)$							
q	0	1	2	3	4	5	6	q	0	1	2	3	4	5	6
0	0	1	2	-	4	5	-	0	0	1	2	3	4	5	6
1	1	1	2	-	5	3	-	1	0	1	2	3	4	5	6
2	2	3	4	2	0	6	-	2	0	1	2	3	4	5	6

Pentru implementare cu $MX_{3,3}$ funcția q^* devine vectorială $q^* = [q_1^* \ q_0^*]$ și se reprezintă în tabelul 2.

$q_1^* = f_1(q_1, q_0, d)$								$q_0^* = f_0(q_1, q_0, d)$							
q ₁	0	0	0	1	1	1	2	q ₁	0	0	0	1	1	1	2
q ₀	0	1	2	0	1	2	0	q ₀	0	1	2	0	1	2	0
d	0	0	0	0	-	1	1	d	0	1	2	-	1	1	1
1	0	0	0	-	1	1	-	1	1	1	2	-	2	1	-
2	0	1	1	0	0	2	-	2	2	0	1	2	0	0	-

Arborele minimal multivalent se da în figura 3.

b) Varianta minimală cu $MX_{2,0}$ și $MX_{2,2}$ este dată în figurile 4 și 5 care reprezintă arborele minimal și schema aferentă, corespunzătoare CLC a generatorului noii stări: $q = [q_2 \ q_1 \ q_0]$, $d = [d_1 \ d_2]$

$$i = [i_1 \ i_0] = [a_1 x_1 y_1 b_1 z_1 a_0 x_0 y_0 b_0 z_0]$$

4.- Concluzii

Metoda de sinteză elaborată înglobează în mod unitar și sistematic atât automatele convenționale binare cât și cele multivalente, indiferent de complexitatea grafului lor de tranziție. Ea are certe avantaje în calculul automat.

BIBLIOGRAFIE

- 1 Haldem, S.A - Binary Decision Diagrams IEEE Trans.on Computers, Vol C-27, nr.6. 1978 pp 509-516
- 2 Postila, Lelia- Procedeu de sinteză automată a circuitelor logice combinate multivalente. Comunicarea Simp. IP Iasi octombrie 1991.

PROCEDEU DE SINTEZA AUTOMATA A CIRCUITELOR LOGICE COMBINATIONALE MULTIVALENTE

Lelia Postila - Institutul Politehnic
Cluj - Napoca

REZUMAT

Metoda propusa se refera la sinteza circuitelor logice combinacionale multivalente utilizand celule functionale de tip multiplexor. Se indica modul de realizare a arborelui de decizie coferent functiei logice date si tipului de celule dorit. Utilizarea arborelui de decizie permite automatizarea procedurii de sinteza si de cautare a unor solutii minimale.

1. Introducere

In ultimii ani s-au extins studiile si realizările legate de logica multivalenta si circuitele larg integrate bazate pe aceasta logica [1]. Acestea sînt circuite care prelucreaza semnale cu mai multe nivele logice. Utilizarea lor permite micșorarea arfel de siliciu si reduce numarul conexiunilor interne si a liniilor de magistrela.

In cele ce urmeaza am utilizat logica multivalenta pentru a dezvolta un procedeu generalizat de proiectare a CLC. procedeu care se aplica atit la sinteza circuitelor logice multivalente cit si a celor binare, care apar ca si cazuri particulare ale primei categorii. Arborele de decizie atasat circuitului cautat este compus din celule functionale de tip multiplexor. Configuratia bazata pe celule functionale este proprie circuitelor larg integrate. Arborele de decizie poate fi creat si minimizat automat permitind analiza mai multor variante pentru cautarea solutiilor optime.

2. Multiplexorul k-valent, celula fundamentala a arborelui de decizie logica.

Multiplexorul k-valent cu r intrari: MX_k^r (fig.1) prezinta urmatoarele caracteristici:

D&

- are r intrări de date și s ieșiri;
- semnalele prelucrate de MX sînt k -valente (cînd $k \geq 2$) circuitul se mai numește $Y [1]$;
- oricare din semnalele Y de intrare poate fi transmis la ieșire selectîndu-l cu un semnal x de adresă, sau de selecție;
- semnalul x poate fi sfîșurător sau temporar, sînd $r \leq k$, adresă se poate forma pe mai multe intrări, prin transpunerea în bază k a fiecărei valori zecimale x^i a semnalului de selecție.

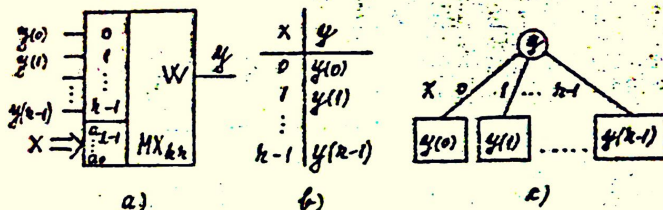


fig. 1.

$$x = [x_{1-1}, x_{1-2}, \dots, x_{0}] = 2^r [1] \quad (1)$$

astfel încît $x^i = s_{1-1}^i, \dots, s_{0k}^i$, $s_1 \in \{0, \dots, k-1\}$,

$$x^i \in \{0, \dots, r-1\}, \quad i = \log_k r$$

Se poate observa că orice multiplexor MX_k realizează funcția logică multivalentă.

$$y = f(x), \quad x: R \rightarrow k, \quad R = \{0, 1, \dots, r-1\}, \quad K = \{0, 1, \dots, k-1\} \quad (2)$$

dată în figura 1, b.

Valoarea funcției y se obține la ieșirea MX. Variabila x are semnificația fizică în cazul în care $r \leq k$ reprezentînd semnalul de adresă. Cînd $r > k$, r este decimul zecimal al adreselor multiplexorului.

Funcția (2) se poate reprezenta prin arborele de decizie din figura 1, c în care în nodurile terminale apar valorile $y(i)$ ale funcției corespunzătoare valorilor $x_i \in R$ ale variabilei x , valori indicate lîngă ramurile arborelui radial.

b) orice multiplexor poate realiza funcția logică de s variabile:

$$y = \sum_{j=0}^{L-1} a_{j-1} \cdot a_{j-2} \cdot \dots \cdot a_0, \quad y: k^L \rightarrow K, \quad n \leq L = \log_k r \quad (3)$$

$a_j, j=0, \dots, L-1$ fiind semnale singulare aplicate la intrările de adresă iar y este ieșirea multiplexorului.

3. Implementarea funcțiilor logice multivalente cu Nx_{kr}

Se pune problema implementării funcției logice multivalente:

$$y = f(x), \quad f: N \rightarrow K, \quad N=0, 1, \dots, n-1, \quad n=0, 1, \dots, m-1 \quad (4)$$

Se va utiliza un multiplexor k valent de r căi Nx_{kr} .

a) Cazul $n \leq r, n \leq k$

Implementarea este directă, conform figurii 1 și cu observațiile făcute la paragraful 2 privind la formarea adresei (variabila x este decodificată în baza k).

b) Cazul $n \leq r, n > k$

Multimea valorilor funcției este mai mare decât numărul nivelelor logice ale semnalelor prelucrate de Nx_{kr} . În acest caz valorile numerice ale funcției se decodifică în baza k , iar funcția y devine vectorială

$$y = [y_{1-1}, y_{1-2}, \dots, y_{1-0}] = [y_1]_1 \quad (5)$$

$$y_j = f_j(x), \quad f_j: N \rightarrow K, \quad 1 \leq j \leq l, \quad l = (\log_k n) + 1, y(i) =$$

$$\begin{bmatrix} f_{1-1} \\ f_{1-2} \\ \vdots \\ f_{1-0} \end{bmatrix} \in K^l$$

Arborele din figura 1, b se descompune pe orizontală în l arbori în paralel, aflați pe același nivel. Fiecare graf $j = 0, \dots, l-1$ avind ca termeni $y_j(i), i = 0, \dots, n-1$.

c) Cazul $n > r, n \leq k$

Cind multimea valorilor variabilei este mai mare decât numărul de intrări ale multiplexorului, fiecare element din multimea N se va reprezenta în baza r , rezultând variabila vectorială.

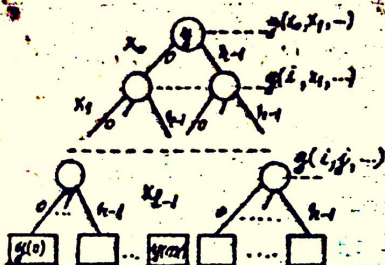
$$x = [x_{1-1}, x_{1-2}, \dots, x_{1-0}] = x_1 = [x_j]_1, \quad 1 \leq j \leq l, \quad l = \log_r n + 1 \quad (6)$$

astfel încât pentru fiecare valoare $x' \in N$, $x' = [x'_{1-1}, x'_{1-2}, \dots, x'_{1-0}]$.

funcția logică devine

$$y = f(x_1, x_2, \dots, x_n), \quad x_i \in \{0, 1\} \quad (7)$$

iar arborele de decizie (figura 2) va descompune pe verticală în l nivele, fiecare conținând un număr de noduri funcționale de tip $M_{k,r}$.



Ca urmare pentru implementarea funcției se vor folosi multiplexoare $M_{k,r}$ conectate în cascadă. Arborele nu este unic și depinde de ordinea de explicitare a variabilelor pe nivele $0, \dots, l-1$, ca urmare vor exista l moduri de implementare posibile.

d) Cazul $n > r$, $m > r$

Se realizează descompunerea pe orizontală și pe verticală a arborelui aferent funcției date, după procedurile descrise mai sus.

4. Etapele proiectării CLC cu multiplexare.

În capitolul 3 s-a arătat că orice funcție logică $y=f(x)$ de forma (4) poate fi implementată cu $M_{k,r}$. Afirmatia se extinde și la funcții de mai multe variabile, caz în care x este un vector de forma (1) sau (6). Etapele proiectării CLC cu $M_{k,r}$ vor fi următoarele:

- se definește funcția sau funcțiile cerute,
- funcția logică se aduce la forma (4) (5) sau (7),
- se construește arborele de decizie aferent acestor forme,
- se reduce arborele prin eliminarea nodurilor redundante.

Condițiile de redundanță sînt îndeplinite dacă:

- valorile terminale ale unui nod funcțional sînt egale între ele,
- două celule funcționale plasate pe același nivel au aceeași

configurație.

Procedurile de reducere sînt iterative și ierarhizate, ca urmare se pot automatiza. Algoritmul de definire a arborelui permite permutarea variabililor sale pe diverse nivele pentru compararea variantelor rezultate în urma minimizării.

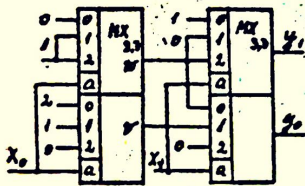
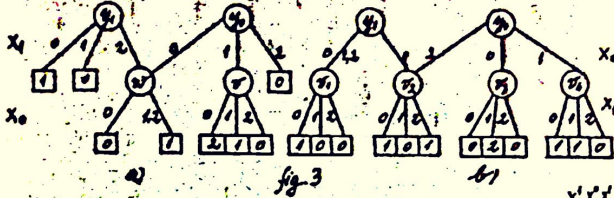


fig. 4

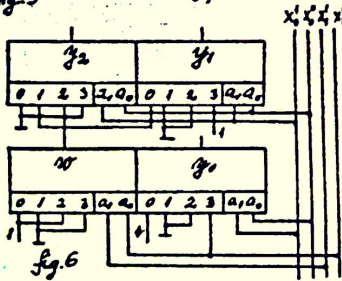


fig. 6

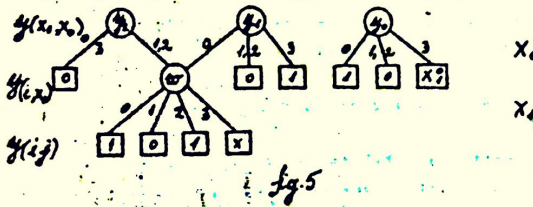


fig. 5

3.- Example

Se va implementa funcția logică $y = f(x)$, $f: M \rightarrow M$, $M = [0, \dots, 8]$; $[M = 0, \dots, 6]$ data în tabelul 1. Se vor utiliza

a) $MUX_{3,2,5}$; b) $MUX_{3,3}$; c) $MUX_{2,4}$

x	0	1	2	3	4	5	6	7	8
y	3	4	4	2	1	0	0	3	3

Tabelul 1

a) Adresa de va exprime prin 2 semnale $[a_1, a_0] = [00, 01, 02, 03, 04, 10, 11, 12, 13]$ iar valorile functiei se optica direct la primele 9 intrari ale multiplexorului.

b) Arborile functiei va avea doua nivele si doua ramuri in paralel, corespunzatoare functiei date in tabelul 2.

x_1	0	0	0	1	1	1	2	2	2
x_0	0	1	2	0	1	2	0	1	2

Tabelul 2

y_1	1	1	1	0	0	0	0	1	1
y_0	0	0	1	2	1	0	0	0	0

x_1	0	0	0	0	1	1	1	1	2
x_0	0	1	2	3	0	1	2	3	0

Tabelul 3

y_2	0	1	1	0	0	0	0	0	0
y_1	1	0	0	1	0	0	0	1	1

y_0	1	0	0	0	1	0	0	1	1
-------	---	---	---	---	---	---	---	---	---

In figurile 5 si 6 se indica variante minimale a arborului, respectiv a schemei rezultate.

5. Concluzii.

Utilizarea logicii multiple largeste aria posibilitatilor de proiectare a CLC cu MX. Procedoul prezentat permite efectuarea unui studiu sistematic pe calculator in vederea stabilirii configuratiei si alegerii solutiilor optime.

B I B L I O G R A F I E

1. Murat, S.L. - Multiple - Valued Logic, Its Status and Its Future. IEEE Trans on Computers, C-33 nr.12 1984.
2. Karpovsky, M - Multilevel logical networks, IEEE Trans. on Computers, C-36, 2, 1987.

1. Introducere

Lucrarea respectiva prezinta o metoda de analiza a sistemelor secventiale care poate fi folosita fie pentru sisteme secventiale clasice fie pentru automate cablate.

Metoda se preteaza la implementarea pe microcalculatoare si consta in stabilirea unei matrici care evidentiaza evolutia starilor sistemului.

Este necesar sa se prezinte cateva notiuni utile:

a). fie conjunctia notata Q de "n" variabile $x_{n-1}, \dots, x_0$

- se defineste exponentul formal - z - pentru o variabila binara

(1)
$$z = \begin{cases} 0 & \text{daca variabila binara } x \text{ apare complementata } (\bar{x}) \\ 1 & \text{daca variabila binara } x \text{ apare necomplementata } (x) \end{cases}$$

- astfel conjunctia poate fi scrisa

(2)
$$Q = \sum x_i^{z_i}$$

b). echivalentul zecimal reprezinta valoarea zecimala a unui numar reprezentat binar (ex. echivalentul zecimal pentru (1011) este 11)

- se completeaza Q cu un indice k care reprezinta echivalentul zecimal al numarului binar format din exponentii formali z_i

(ex. pentru $Q = \bar{x}_2 x_1 \bar{x}_0 \Rightarrow Q = x_2^0 x_1^1 x_0^0 \Rightarrow k = (010) = 2 \Rightarrow Q_2 = x_2^0 x_1^1 x_0^0$)

ceea ce permite identificarea fiecarei conjunctii prin indicele atasat.

2. Analiza sistemelor secventiale

Fie un sistem secvential cu "p" stari si "q" intrari, descris de n ecuatii de stare

(3) $y_i = f_i(x_{q-1}, \dots, x_0, y_{n-1}, \dots, y_0) \quad i = 0, n-1$

Nota: Textele din chenar reprezinta indicatii si exemple

unde f_1 reprezintă o sumă de conjuncții.

Dar o conjuncție din care lipsește cel puțin o variabilă (formă minimizată) poate fi scrisă ca o sumă de conjuncții elementare (ex. fie $n = 2$ și $Q = \bar{y}_2 y_0 \Rightarrow Q = \bar{y}_2 \bar{y}_1 y_0 + \bar{y}_2 y_1 y_0$).

Astfel, ecuația de stare poate fi scrisă:

$$(4) \quad x_1 = \sum_i (g_{1i}(x_{q-1}, \dots, x_0) \cdot (\sum_k Q_k))$$

(ex: dacă $y_2 = x y_1 \bar{y}_0 + \bar{x} y_2 y_1 + y_2 \bar{y}_1 y_0$ atunci $y_2 = x(\bar{y}_2 y_1 \bar{y}_0 + y_2 y_1 \bar{y}_0) + \bar{x}(y_2 y_1 \bar{y}_0 + y_2 y_1 y_0) + y_2 \bar{y}_1 y_0$ unde $g_{21} = x, g_{22} = \bar{x}, g_{23} = 1$)

În expresia (4) o conjuncție Q_k poate să apară de mai multe ori însoțită de funcții g_{ij} diferite. Prin gruparea termenilor Q_k se obține pentru y_j expresia generală:

$$(5) \quad x_1 = \sum_k a_{1k} Q_k$$

Dacă se consideră sistemul cu următoarele ecuații:

$$(5.1) \quad \begin{cases} x_2 = \bar{y}_2 \bar{y}_1 x + \bar{y}_2 \bar{y}_1 y_0 + y_2 y_1 \bar{x} + \bar{y}_1 \bar{y}_0 x \\ x_1 = \bar{y}_2 y_1 \bar{y}_0 + \bar{y}_2 y_0 \bar{x} \\ x_0 = \bar{y}_2 y_0 + y_1 \bar{y}_0 \bar{x} + \bar{y}_1 \bar{y}_0 \bar{x} \end{cases}$$

atunci:

$$(5.2) \quad \begin{cases} x_2 = y_2^0 y_1^0 y_0^0 x + y_2^0 y_1^1 y_0^0 x + y_2^0 y_1^0 y_1^1 + y_2^1 y_1^0 y_0^0 x + y_2^1 y_1^0 y_1^0 x + y_2^1 y_1^1 y_0^0 x + y_2^1 y_1^1 y_1^1 x \\ x_1 = y_2^0 y_1^0 y_0^1 x + y_2^0 y_1^1 y_0^0 + y_2^0 y_1^1 y_0^1 x \\ x_0 = y_2^0 y_1^0 y_0^0 + y_2^0 y_1^1 y_0^0 + y_2^1 y_1^0 y_0^0 x + y_2^1 y_1^1 y_0^0 x \end{cases}$$

sau:

$$(5.3) \quad \begin{cases} x_2 = Q_0 + x Q_1 + Q_3 + x Q_4 + \bar{x} Q_5 + \bar{x} Q_6 + \bar{x} Q_7 \\ x_1 = \bar{x} Q_1 + Q_2 + \bar{x} Q_3 \\ x_0 = Q_0 + Q_2 + x Q_4 + \bar{x} Q_6 \end{cases}$$

Se trece ușor de la ecuațiile (5) la o ecuație matricială

$$(6) \begin{bmatrix} x_{n-1} \\ x_{n-2} \\ \vdots \\ x_1 \\ x_0 \end{bmatrix} = \begin{bmatrix} k_{n-1,0} & k_{n-1,1} & \dots & k_{n-1,n-1} \\ k_{n-2,0} & k_{n-2,1} & \dots & k_{n-2,n-1} \\ \vdots & \vdots & \ddots & \vdots \\ k_{1,0} & k_{1,1} & \dots & k_{1,n-1} \\ k_{0,0} & k_{0,1} & \dots & k_{0,n-1} \end{bmatrix} \cdot \begin{bmatrix} z_0 \\ z_1 \\ \vdots \\ z_{n-2} \\ z_{n-1} \end{bmatrix} \quad \text{sau } i=n-1$$

unde:

$$k_{ij} = \begin{cases} b_{ij} & \text{dacă } j \in Y_i \\ 0 & \text{dacă } j \notin Y_i \end{cases}$$

Matricea K pune în evidență succesiunea stărilor; starea curentă este dată de indicele unei coloane, iar starea următoare este dată de echivalentul zecimal al numărului binar format din elementele coloanei (ex: din starea 0 (S_0) sistemul va evoluia spre starea S_n , unde: $n = (k_{n-1,0} \ k_{n-2,0} \ \dots \ k_{1,0} \ k_{0,0})$).

Pentru cazul particular caracterizat de ecuațiile (5.7) se obține ecuația matricială:

$$(6.1) \begin{bmatrix} x_2 \\ x_1 \\ x_0 \end{bmatrix} = \begin{bmatrix} x & x & 0 & 1 & x & \bar{x} & \bar{x} & \bar{x} \\ 0 & \bar{x} & 1 & \bar{x} & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & x & 0 & \bar{x} & 0 \end{bmatrix} \cdot \begin{bmatrix} z_0 \\ z_1 \\ \vdots \\ z_7 \end{bmatrix}$$

$$x = 0 \rightarrow \begin{bmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \quad ; \quad x = 1 \rightarrow \begin{bmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}$$

Obs: deasupra primei linii din matricea A s-au trecut stările curente.

Din starea 0 - S_0 , sistemul va evoluia spre starea dată de $(x \ 0 \ 1)$, iar din starea 3 - S_3 spre starea dată de $(1 \ x \ 0)$; pentru $x = 0$: $S_0 \rightarrow S_3$ și $S_3 \rightarrow S_6$, iar pentru $x = 1$: $S_0 \rightarrow S_5$, S_1 , $S_3 \rightarrow S_4$.

Pe baza matricei K se poate stabili graficul sistemului (fig. 1).

Metoda se aplică similar și pentru sisteme secvențiale cu mai multe intrări, situație în care elementele k_{ij} vor fi funcții de

mai multe variabile.

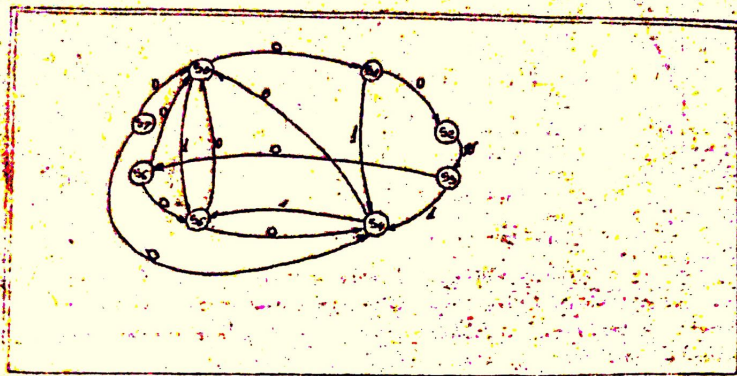


Fig. 1.

3. Implementarea algoritmului

Fie un sistem secvențial caracterizat de n ecuații de stare (n variabile de stare) asupra cărora se aplică metoda prezentată.

Implementarea necesită rezervarea unui cuvânt pentru fiecare conjuncție din cei $(2^n - 1)$ conjuncții elementare care conțin exponenții formali ai conjuncțiilor, începând cu Q_0 și terminând cu $Q_{2^n - 1}$. Fiecărei conjuncții i se aplică o variabilă P_k (inițial 0), care reprezintă de fapt termenul k_{ij} . La introducerea unei conjuncții din expresia variabilei de stare X_{ij} se generează două cuvinte de n biți: primul are poziționați biții corespunzători variabilelor de stare care se găsesc în conjuncție, iar al doilea cuvânt exponenții formali ai variabilelor de stare. Aceste două cuvinte sînt folosite pentru toate conjuncțiile din expresiile variabilelor de stare. Se înmulțește logic primul cuvînt cu cuvintele corespunzătoare celor $2^n - 1$ conjuncții elementare și se compară rezultatul cu al doilea octet (referința (2)); în caz de identitate se succedă P_i cu funcția G , care înmulțește conjuncția (rel. (4)); dacă $G = 1$ se consideră $P_j = 1$ și nu se mai execută alte sumări.

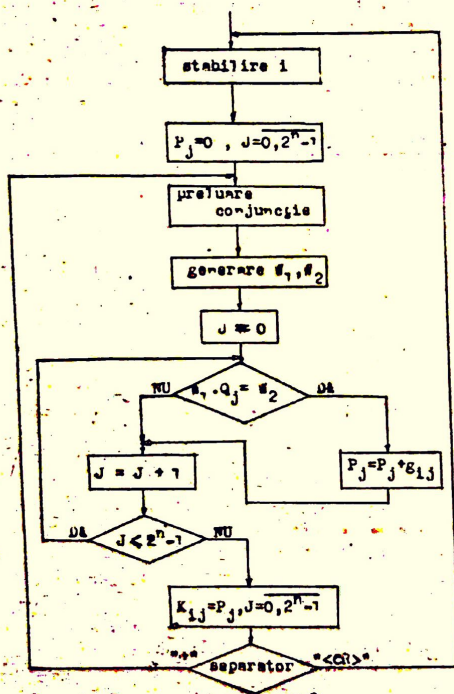
Operațiile se repetă pentru fiecare conjuncție din expresia lui X_i , după introducerea acestuia, înalt după introducerea ecuației linia corespunzătoare din matrice este completă și se

poate prelua din zona P; acest lucru permite ca zonele de memorie de lucru să fie utilizate pentru următoarea ecuație

Dacă vectorul de intrare conține mai multe variabile la introducerea acestora se stabilește ușor dacă un element al termenilor k_{ij} este 1 prin compararea exponenților formali cu valorile particulare (fie $Q_k = \prod x_{ij}^{s_{kj}}$, atunci $Q_k = 1 \Leftrightarrow x_{ij} = s_{ij}$)

Se consideră ca separatori caracterele funcționale " = ", " + ", " < CR > ",

Organigrama de principiu este prezentată în fig. 2.



Se exemplifică metoda pentru sistemul dat de ecuațiile (5.1)

- în memorie există tabelul:

	0 0 0
	0 0 1
	0 1 0
(6.2)	0 1 1
	1 0 0
	1 0 1
	1 1 0
	1 1 1

- după introducerea primei conjuncții separată de operatorul
 $\wedge$, ($x(y_2 y_1)$) se construiesc cele două cuvinte de conținut
 $(1, 1, 0)$ și $(0, 0, 0)$; prin algoritmul prezentat referința (2); se
 identifică conjuncțiile Q_0 și Q_1 - $k_{2,0} = 1$, $k_{2,1} = x \cdot y \cdot a.m.d.$
 conform tabelului:

conjuncție X_2	cuvânt 1	cuvânt 2	$x_1 \cdot Q_j = x_2$	I	$k_{1,1}$
$x(y_2 y_1)$	1 1 0	0 0 0	Q_0	0	$P_0 = x, P_1 = x$
			Q_1	1	
$\bar{x}(y_2 y_1)$	1 1 1	0 1 1	Q_3	3	$P_3 = \bar{x}$
$\bar{x}(y_2 \bar{y}_1)$	1 1 0	1 1 0	Q_6	6	$P_6 = \bar{x}, P_7 = \bar{x}$
			Q_7	7	
$\bar{x}(y_2 \bar{y}_1)$	1 0 1	1 0 1	Q_5	5	$P_5 = \bar{x}, P_7 = \bar{x} + \bar{x} = \bar{x}$
			Q_7	7	
$x(\bar{y}_1 \bar{y}_2)$	0 1 1	0 0 0	Q_0	0	$P_0 = x + x = x, P_4 = x$
			Q_4	4	

Tab. 1

- elementele $k_{2,j}$ se preiau din zona P (variabilele $P_0, \dots, P_7$)
 $(k_{2,0}, k_{2,1}, k_{2,3}, k_{2,6}, k_{2,5}, k_{2,7}) = (x, x, \bar{x}, x, \bar{x}, \bar{x})$
- ultima conjuncție este separată de caracterul <CR>
- Se pot genera apoi cele 2 valori ale variabilei x , obținându-se
 cele 2 matrici derivate din K , deci toate stările și tranzițiile
 sistemului secvențial.

Pentru mai multe variabile de intrare, sistemul poate cere
 o anumită configurație a vectorului de intrare, determinând
 evoluția stărilor.

Se exemplifică metoda pentru sistemul dat de ecuațiile (5.1)

- în memoria există tabela:

	0 0 0
	0 0 1
	0 1 0
(6.2)	0 1 1
	1 0 0
	1 0 1
	1 1 0
	1 1 1

- după introducerea primei conjuncții separată de operatorul $\wedge$, ($x_1 y_2 y_1$) se construiesc cele două cuvinte de conținut $\{1, 1, 0\}$ și $\{0, 0, 0\}$; prin algoritmul prezentat referința (2); se identifică conjuncțiile Q_0 și Q_1 - $k_{2,0} = 1$; $k_{2,1} = 1$ g.s.m.d. conform tabelului:

conjuncție I_2	cuvint 1	cuvint 2	$x_1, Q_1 = x_2$	I	$k_{1,1}$
$x_1 \bar{y}_2 \bar{y}_1$	1 1 0	0 0 0	Q_0	0	$P_0 = x_1, P_1 = x_1$
			Q_1	1	
$\bar{y}_2 y_1 y_0$	1 1 1	0 1 1	Q_3	3	$P_3 = y_1$
$\bar{x}_1 (y_2 x_1)$	1 1 0	1 1 0	Q_6	6	$P_6 = \bar{x}_1, P_7 = \bar{x}_1$
			Q_7	7	
$\bar{x}_1 (y_2 y_1)$	1 0 1	1 0 1	Q_5	5	$P_5 = \bar{x}_1, P_7 = \bar{x}_1 + \bar{x}_1 \bar{x}_1$
			Q_7	7	
$x_1 (\bar{y}_1 \bar{y}_0)$	0 1 1	0 0 0	Q_0	0	$P_0 = x_1 + x_1 x_1, P_1 = x_1$
			Q_4	4	

Tab. 1

- elementele $k_{2,j}$ se preiau din zona P (variabilele $P_0, \dots, P_7$)
($k_{2,0}, k_{2,1}, k_{2,3}, k_{2,4}, k_{2,5}, k_{2,6}, k_{2,7}$) = $(x, x, 0, \bar{x}, x, \bar{x}, \bar{x})$
- ultima conjuncție este separată de caracterul <CR>
- Se pot genera apoi cele 2 valori ale variabilei x , obținându-se cele 2 metrici derivate din Δ , deci toate stările și tranzițiile sistemului asociației.

Pentru mai multe variabile de intrare, sistemul poate crea o nouă configurație a vectorului de intrare, determinând evoluția stărilor.

Similar se poate proceda pentru vectorul de ieşire Z . Presupunem că sistemul secvenţial are n ieşiri: z_{n-1}, z_0 . Similar ca pentru variabile de stare, folosind aceeaşi metodă, se ajunge la relaţia (7).

$$(7) \begin{bmatrix} z_{n-1} \\ z_{n-2} \\ \vdots \\ z_1 \\ z_0 \end{bmatrix} = \begin{bmatrix} u_{n-1,0} & \dots & u_{n-1,2^{n-1}} \\ u_{n-2,0} & \dots & u_{n-2,2^{n-1}} \\ \vdots & & \vdots \\ u_{1,0} & \dots & u_{1,2^{n-1}} \\ u_{0,0} & \dots & u_{0,2^{n-1}} \end{bmatrix} \cdot \begin{bmatrix} q_0 \\ q_1 \\ \vdots \\ q_{2^n-1} \end{bmatrix}$$

unde elementele de pe coloana "i" reprezintă valorile variabilelor de ieşire corespunzătoare stării "i".

După introducerea ecuaţiilor de ieşire, programul afişează stările codificate prin numărul lor şi valorile corespunzătoare ale variabilelor Z .

Ca exemplu, considerînd că pentru sistemul descris de ecuaţiile (5.1) ieşirile sînt:

$$(7.1) \begin{cases} z_1 = y_2 y_0 + y_2 y_0 \\ z_0 = y_0 \end{cases}$$

se obţine:

$$(7.2) \begin{bmatrix} z_1 \\ z_0 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} q_0 \\ q_1 \\ \vdots \\ q_7 \end{bmatrix}$$

- pentru stare 3 - ieşirea va fi $(z_1, z_0) = (1, 0)$

Procesarea ecuaţiilor de ieşire are loc după aceeaşi metodă ca la ecuaţiile de stare.

Ca şi în referinţa (2) sistemul poate fi prevăzut cu o interfaţă paralelă, devenind un sistem - expert de simulare a sistemelor secvenţiale.

4. Concluzii

Fără de alte metode - referinţa (2), această metodă prezintă următoarele avantaje:

- o zona relativ mica de memorie - cite un cuvint pentru fiecare conjunctia Q_k din cele $2^n - 1$ conjunctii elementare;
- zona de memorie corespunzatoare conjunctiilor dintr-o ecuatie este comuna, in sensul ca este folosita pentru toate ecuatiile, deoarece dupa introducerea unei ecuatii se poate determina complet linia corespunzatoare din matricea K (eventual pe durata introducerii urmatoarei linii)
- timp de executie foarte mic, din punct de vedere utilizator, deoarece o linie se calculeaza imediat de ecuatia de stare a fost introdusa.
- raspuns rapid dupa introducerea unui vector de intrare deoarece expresiile elementelor k_{ij} au fost stabilite anterior

Bibliografie

- | | |
|-------------------------|--|
| 1. Al.Valachi, M.Birsan | - Tehnici numerice si Automate, Ed.Junimea, 1987 |
| 2. I.Dancea | - Dynamically changing the Logical Behavior of a Microcomputer Interface |
| 3. Vangalur S.Alagar | - Fundamentals of computing, Prentice Hall, 1989 |
| 4. H.Taub | - Digital circuits and microprocessors McGraw-Hill, 1985 |
| 5. C.R.Clare | - Designing logic systems using states machines McGraw-Hill, 1972 |
| 6. G.Russell | - Computer aided tools for VLSI system desing, Peter Peregrinnsltd, 1987 |

UTILIZAREA ÎNCREDINȚĂ A TIMERULUI 8253 ÎN COMUNICĂȚII SERIALE

prof. dr. ing. Al. Valachi
I.P. Iasi
asist. ing. Dan Olaru
I.P. Iasi

1. Introducere

Cele mai utilizate metode de realizare a comunicății seriale, mod de transmisie a datelor numerice cu evidente avantaje, sînt :

a) folosirea a doua porturi (unul de intrare iar celălalt de ieșire) și efectuarea de către microprocesor a operațiilor de serializare/deserializare. Această metodă, cea mai economică din punct de vedere hardware, prezintă dezavantajul utilizării ineficiente a microprocesorului și este folosită cu precădere în sistemele dedicate ieftine.

b) utilizarea unui circuit VLSI dedicat (8251, 280-SIO, etc.). În continuare vom propune o a treia variantă de implementare a comunicății seriale prin folosirea unui circuit timer (8253, 280-CTC). Metoda propusă asigură o utilizare superioară a mp comparativ cu varianta a), în absența unui circuit de tip USART, și permite eliminarea circuitelor pentru generarea tactului de comunicație. Circuitul timer poate fi folosit în același timp și pentru alte aplicații tipice ale acestuia.

2. Principiul metodei

Vom considera o linie de comunicație semiduplex cu viteza de 1200 bauds conectată la un microsistem cu frecvența ceasului de 2 Mhz. Transmisia este de tip asincron : 1 bit START, 8 biti date, 1 bit STOP, fără control de paritate. Modalitatea de conectare a circuitului 8253 [2] este cea din figura 1.

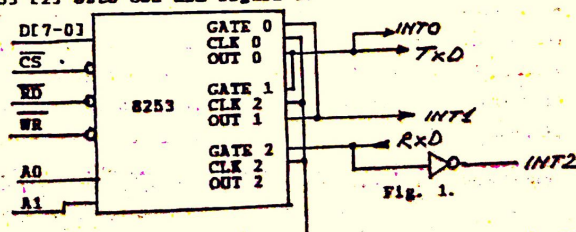


Fig. 1.

Iesirea OUT 0 (numaratorul #0) este conectata la linia de emisie TxD, iar intrarea GATE 2 (numaratorul #2) la linia de receptie RxD. Daca f este frecventa de transmitie si F este ceasul procesorului din sistem, vom nota cu $n0=F/f$ numarul de perioade de ceas procesor cuprinse in intervalul necesar transmisiei/receptiei unui bit. Selectarea unui numarator se face prin transmiterea cuvintului de control corespunzator (fig. 2)(21).

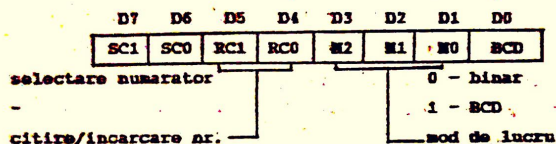


Fig. 2.

Pentru emisie se va utiliza numaratorul #0 in vederea stabilirii duratei Low a semnalului pe linie si numaratorul #1 pentru durata High a aceluasi semnal (TxD). Se vor programa nr. #0 in modul 1(programmable one-shot) iar nr. #1 in modul 4 (software triggered strobe), inainte de inceperea propriu-zisa a emisiei, si apoi in modul 1. In vederea receptiei numatoarele se vor programa dupa cum urmeaza : nr. #2 in mod 4, nr. #1 in mod 1, nr.#0 in mod 4. In cele ce urmeaza vom considera pe rand cele doua situatii.

2.1 Receptia

Procedura de receptie este declansata de primul front negativ al semnalului RxD. Esantionarea liniei RxD este realizata prin intreruperi succesive generate de nr. #1 (INT1). Organigrama procesului de receptie este ilustrata in figura 3.

La primul front negativ al semnalului RxD se genereaza o intrerupere (INT2). In rutina de tratare a acestei intreruperi se mascheaza INT2, se incarca nr.#2 cu $Rmax>8*n0$ si nr. #1 cu $n0/2$ (durata unei jumatai de bit). Deserializarea se realizeaza prin

determinarea succesivă a bitilor $CH[i]$. Inițial $CH[0]=0$ și variabila contor $I=-1$. După anularea continutului nr. 31 se generează INT1. Dacă $I=-1$ și continutul nr. 2 este diferit de H_{max} rezultă că nu a fost vorba de un bit de start (pe durata $T_{bit}/2$ a revenit în starea High). În această situație se demarchează întreruperile și se revine în programul principal. Dacă în schimb continutul nr. 32 nu este modificat, se va reincarca nr. 31 cu n0 și se revine în programul principal. În situația $I=-1$ se prelevează continutul nr. 32 calculându-se $n2$ (variația continutului față de situația precedentă). În funcție de valoarea $n2$ se determină $CH[i]$. Procesul continuă pînă se determină toți cei 8 biti de date și bitul de STOP.

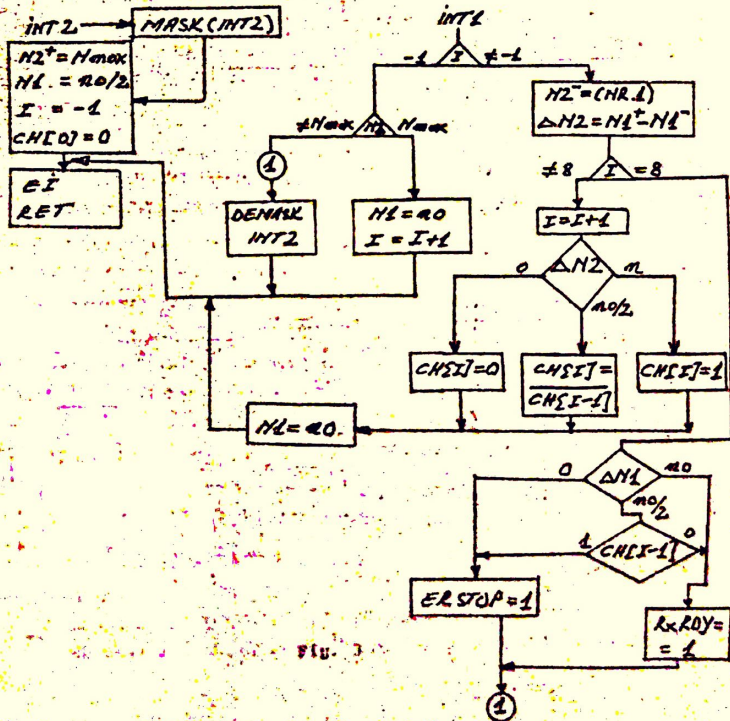


Fig. 3

010

unda pentru exemplul considerat.

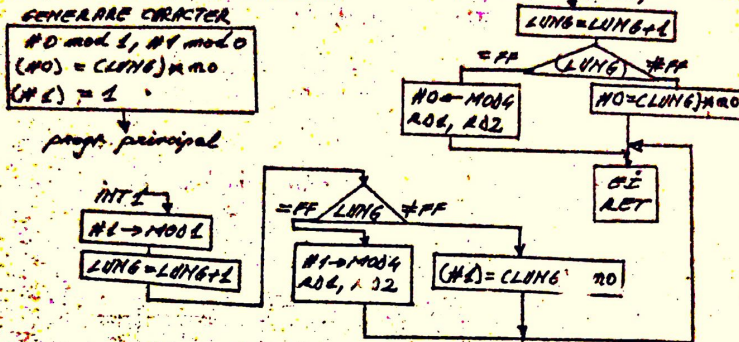


Fig. 5c

Initial se programeaza numaratoarele #0 si #1 in modul 1, respectiv 0, dupa care se incarca nr. #0 cu $n0 * L$ (L fiind prima valoare din stiva LUNG) si nr. #1 cu 1. Dupa un timp egal cu perioada ceasului 0 se obtine un front pozitiv la iesirea INT1. Tratarea acestei intreruperi consta in incrementarea variabilei LUNG si testarea continutului locatiei (LUNG). Daca acesta este diferit de FF, se reprogreamaza nr. #1 in modul 1 si se incarca cu valoarea $E * n0$. La frontul pozitiv al semnalului TxD (INT1) se activeaza nr. #1, se blocheaza nr. #0 si apoi se incarca cu noua constanta $L * n0$. Procesul continua pina se intalneste sfirsitul caracterului (FF in buffer), moment in care se trece numaratorul respectiv in modul 4. In acest mod se asigura starea High la iesire si se blocheaza functionarea astabila a grupului format din nr. #0 si nr. #1.

Bibliografie

- Bibliografie**
[1] Th.R. Blakeslee - Proiectarea cu circuite logice MSI si LSI
Ed. Tehnica - 1988
[2] *** - Catalog INTEL 1985

SIMULARE GRAFICA A UNUI PLANIFICATOR DE ACTIUNI PENTRU ROBOTI

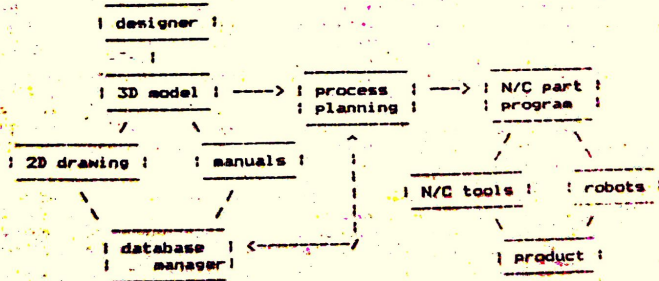
ing. S. Ticrea A.C.D.M.I.N. Cluj
asist. ing. C. Rusu I.P. Cluj

1. PROBLEMA ABORDATA

Obiectul lucrării de față îl constituie realizarea unui produs program în domeniul roboticii axat pe generarea de planuri de acțiuni și modelarea lumii exterioare pentru un robot din lumea cuburilor.

Tema planificării robotilor a fost abordată încă din anii '70 prin proiectele universităților M.I.T. și Stanford și care treptat a început să preocupe mai toate institutele de cercetări din întreaga lume. Au fost realizate astfel așa numitele "robot planning systems" printre care se numără PLANNER, DCOMP, RSTRIPS și altele, sisteme bazate pe calculul predicatelor și pe teoria cadrelor.

Pornind de la aceste considerente și de la rezultatele obținute în acest domeniu, am considerat utilă realizarea unui planificator de roboti inteligent într-un sistem grafic de simulare pe care l-am denumit RSRP (Romanian System of Robot Planning). Acest produs se dorește a fi o primă componentă a unui sistem CAD-CAM destinată cercetării și testării diferitelor tipuri de ansambluri robot capabile de a găsi singure secvența de acțiuni pentru atingerea unui anumit scop. Un sistem CAD-CAM are în general o structură de tipul celei descrise în schema bloc de mai jos :



Se observa clar delimitarea si interactiunea celor patru module ce alcatuiesc acest sistem: modulul CAD, planificatorul propriu-zis, baza de cunostinte a sistemului si modulul CAM. Lucrarea de fata descrie implementarea primelor trei module in domeniul lunei cuburilor abordind probleme de CAD, infografica, inteligenta artificiala si calcul al predicatelor. Totodata a fost realizat un produs soft independent destinat interfetarii sistem-utilizator in conditiile grafice oferite de calculatorul utilizat.

Sistemul creat astfel are urmatoarele facilitati:

- posibilitatea reprezentarii in trei (3) dimensiuni a mediului
- posibilitatea construirii interactive a lunei de cuburi impreuna cu attributele sale
- facilitati de salvare/restaurare a scenei
- posibilitatea elaborarii unui plan de actiuni (descrie pe baza unui limbaj formal) pentru atingeres unei anumite situatii pozitionale a cuburilor.

2. MODULUL C.A.D.

2.1. Alegerea modelului de reprezentare

Prima problema abordata in modul a fost gasirea modelului de reprezentare a lunei inconjuratoare. Modelarea mediului si manipularea acestui model reprezinta cheia modulului CAD al oricarui produs program. Pe de o parte un model trebuie sa fie astfel ales incit sa asigure o mare flexibilitate in manipulara sa. Pe de alta parte trebuie astfel ales incit sa inglobeze o cantitate suficient de mare de informatii pentru a reprezenta obiectele din jur cit mai exact. Am ales pe baza acestor considerente modelul hybrid-edges (1988) [3]. Modelul sistemului este structurat pe doua nivele astfel:

- pe primul nivel se plaseaza obiectele primare. Aceste obiecte se afla descrise intr-un sistem propriu de coordonate, codificat pe suportul fizic extern (disketta). Ele alcatuiesc astfel biblioteca de componente a sistemului avind posibilitatea astfel de a o extinde cu noi tipuri de obiecte.
- pe al doilea nivel se plaseaza instantele obiectelor primare ce reprezinta de fapt obiectele propriu-zise din jurul nostru. Instantele contin attributele ce individualizeaza instanta si o legatura la obiectul primar ce a generat-o.

Pentru actualizarea permanentă a nivelului doi al modelului au fost implementați operatori de "creare" și "stergere" ce permit crearea, respectiv ștergerea, unei instanțe.

2.2. Reprezentarea 2D

Modelul de reprezentare ales se bazează pe utilizarea unui sistem de coordonate rectangular de tip RNS. Pentru obținerea unei viziuni tridimensionale asupra scenei a fost folosită proiecția paralelă după o direcție, pe planul XOY. Sistemul fiind interactiv, există posibilitatea deplasării pe toate cele trei axe de coordonate prin intermediul unui cursor grafic tip săgeată. În permanentă este afișată poziția reală din "world coordinate" (WC) în care ne situăm. Proiecția unui corp a ținut cont de existența fetelor ascunse eliminând astfel calcule în plus. De asemenea proiecția respectă algoritmul "pictorului", obiectele fiind proiectate începând de la coordonata z cea mai mică spre cea mai mare. Pentru o cit mai bună viziune, am asigurat posibilitatea detalierei unor părți din scena prin facilități de zoom oricât de adânc, revenirea la viziunea de ansamblu a scenei realizându-se urmând drumul invers detalierei, sau direct. Aceste facilități s-au obligat la realizarea și a clippingului 3D a scenei după metoda Cohen Sutherland [4].

3. BAZA DE CUNOSTINTE GLOBALĂ

După cum este lesne de presupus după cele prezentate în capitolele anterioare, această bază de cunoștințe trebuie să conțină atât informații analitice cit și formale despre lumea exterioară. Prin urmare această bază de cunoștințe a fost concepută sub forma unei liste de obiecte individualizate prin atributele sale. Astfel un obiect are un nume, greutate, culoare și informații formale privitoare la poziția acestuia în raport cu celelalte obiecte. Analitic, fiecare obiect este bine determinat prin poziționarea precisă în WC a originii acestuia din sistemul cartezian propriu. Aceste informații fac parte din baza de cunoștințe dinamică datorită operațiilor de creare/ștergere. Tot din baza de cunoștințe dinamică mai face parte și descrierea prin predicate formale a stării curențe în timpul procesării planificatorului.

Cealaltă parte a bazei de cunoștințe o reprezintă cea statică. Din ea fac parte pe de-o parte descrierile stării inițiale și a celei finale (scop) sub aceeași formă cu cea curență, iar pe de altă parte descrierea regulilor folosite de mecanismul de planificare. În fine, baza de cunoștințe statică mai conține un tabel în care fiecărui predicat formal ce caracterizează relația între obiecte i se atribuie un cod folosit în prelucrările ulterioare de către analizorul sintactic.

4. PLANIFICATORUL PROPRIU-ZIS

4.1. Descrierea formală

Multe din facilitățile oferite de sistemele ce imită inteligența artificială implică soluții ale unor probleme bazate pe planuri de acțiuni și luarea unor decizii într-un timp scurt. În domeniul roboticii, problemele tipice care se pun sînt de fapt obținerea scopurilor prin secvențe de acțiuni. Aceste secvențe vor sta la baza programării implicite ulterioare a robotului prin executia efectivă a mișcărilor. Prin acceptiunea de "problema robot" înțelegem un robot care are de îndeplinit acțiuni într-un univers simplificat. Caracteristica principală a unui limbaj implicit constă în aceea că aceste "acțiuni de îndeplinit" descriu acțiunile pe care robotul trebuie să le îndeplinească și nu mișcările necesare pentru realizarea acestor acțiuni.

În lumea cuburilor universul robotului îl reprezintă cîteva cuburi denumite situate pe o masă, sau unul peste altul și dintr-un robot dotat cu o mîna ce prinde și mută corpuri. Sintetizarea acțiunii pentru robot poate fi soluționată printr-un sistem de producție axat pe o bază de date globală care să oglindiască situația universului din care robotul face parte și regulile ce implică acțiunile. O astfel de regulă are în general forma:

$$C1 \wedge C2 \wedge \dots \wedge C_n \longrightarrow A1 \wedge A2 \wedge \dots \wedge A_n$$

unde C_i - condiții și A_j - descrieri de acțiuni care se considera a fi adecvate de executat în situațiile descrise de reguli. Datele astfel reprezentate sînt apoi manipulate de către un interpretor care compară condițiile fiecărei reguli (productii) cu descrierea situației curente pînă cînd aceasta va identifica scopul propus. Se observă astfel că sistemul de producție oferă un control flexibil al operațiilor.

În cazul de față descrierea stărilor și a scopurilor poate fi făcută prin intermediul predicatelor. Se definesc astfel pentru lumea cuburilor predicatele:

CLEAR(x),
ON(x,y),
ONTABLE(x),
HOLDING(x),
HANDEEMPTY.



Descrierea unei stări de moment se bazează tot pe formulele logice predicative. Spre exemplu situația de mai sus poate fi descrisă prin conjunctia predicatelor

$$ON(a,b) \wedge ON(b,c) \wedge ONTABLE(c) \wedge CLEAR(a) \wedge HANDEEMPTY.$$

Avînd aceste considerente de bază am ales ca reguli în sistemele de producție regulile STRIPS-ului [1,2], care au trei componente:

- lista **pre**: conjunctie de predicate reprezentind conditiile
- lista **delete**: conjunctie de predicate ce nu vor mai fi adevarate dupa aplicarea regulii
- lista **add**: conjunctie de predicate ce vor devenii adevarate.

Strategia de cautare abordata este una de tip backward in care conditia aplicabilitatii regulii este aceea de a exista in expresia situatiei curente un predicat ce are corespondent in unul din predicatele listei "add" a regulii. Noua stare curenta este creata prin regresia celorlalte predicate din fosta stare curenta prin instantierea regulii F, conjugandu-le cu aceasta. (Definirea formalismului regresiei este prezentat in [23]).

4.2. Mod de abordare

Descrierea starii finale (scop) cade asadar in sarcina utilizatorului care trebuie sa furnizeze lista de conjunctii de predicate ce caracterizeaza aceasta stare. Pe baza predicatelor introduse in urma unei analize sintactice si semantice este creata sub aceeasi forma lista starii curente si cea a starii initiale care se preia din situatia existenta creata interactiv de utilizator. Algoritmul planificatorului se bazeaza pe constructia unei stive de scopuri continind predicate scop ce se obtin treptat pe masura ce algoritmul backward avanseaza. Aceasta stiva scop se construiesc astfel:

- se pleaca de la unul din predicatele din scopul compus ce regresat printr-o regula la "true".
- conform modalitatii de esire a subscopului se vor adauga pe stiva in plus regula impreuna cu predicatele din lista "pre" a regulii B gasite. O astfel de adaugare la stiva reprezinta de fapt crearea unei derivatii pe stiva, derivatie care va incerca sa rezolve predicatul scop aflat la baza ei. Noile predicate scop obtinute se vor trata in acelasi mod, considerindu-se un predicat ca fiind rezolvat atunci cind toate predicatele de sub el (stiva crescind in jos) au fost la rindul lor rezolvate, sau in cazurile in care predicatele din derivatia sa apartin, toate, starii initiale. Determinarea regulii B rezolvitoare se face prin baleierea de catre procesorul de inferenta a setului de reguli implementat. O regula B este gasita rezolvitoare cind sint indeplinite anumite conditii dintre care cele mai importante ar fi:
 - * predicatul regreseaza "true" prin regula
 - * predicatele (restul) din lista "add" exista in descrierea starii curente dar nu si in cea initiala
 - * predicatele din lista "pre" nu exista printre cele in curs de rezolvare pe nivelele inferioare ale stivei.

Exista insa posibilitatea ca nici o regula sa nu rezolve predicatul analizat. Aici intervine "cheia" algoritmului: posibilitatea transferului pe pozitia predicatului nerezolvat a unui alt predicat, asa numit "ajutator", din starea finala si care devine noul predicat scop. Exista doua posibilitati:

1) in urma rezolvarii acestui predicat revenirea la predicatul initial si regresia acestuia sa ofere satisfactie

2) regresia sa dea "fals" caz in care se incearca un nou transfer. In cazul in care nu mai este nimic de transferat derivatia se sterge de pe stiva si se continua analiza predicatului de la baza derivatiei din locul de unde s-a ramas. De mentionat ca algoritmul este capabil sa restaureze contextul unei derivatii, nu prin apel recursiv, ci repetitiv.

Algoritmul se termina cind predicatelor din conjunctia starii finale sint rezolvate in totalitate. In caz contrar planificarea este imposibila existind prea putine informatii referitoare la starea finala.

5. SISTEMUL DE MENIURI GRAFICE UNIVERSAL

Produsului i s-a atasat un sistem de meniuri grafice de tip pop-up, pull-down dupa modelul Borland. Acest sistem este capabil sa isi gestioneze singur resursele ecran si in plus constructia meniurilor se realizeaza pe baza unui fisier furnizat de utilizator. Acest fisier contine codificat arborele de meniuri intr-o parcurgere depth-first, coltul stinga sus al primei comenzi pop-up si latimea meniului pop-up. Sistemul este livrat sub forma de fisier .OBJ cu posibilitatea "legarii" acestuia la aplicatia dorita.

6. CONCLUZII

Lucrarea abordeaza problematica generarii planului de actiune a unui robot in lumea cuburilor cautind solutia optima de atingere a unui scop final pornind de la o stare initiala data. Pachetul de programe este bazat pe o grafica interactiva ce permite utilizatorului descrierea starii initiale, cele finale fiind specificate cu ajutorul predicatelor formale. Acest program a fost implementat in limbajul C pe microcalculatoare compatibile IBM-AT, obtinindu-se rezultate de genul celor prezentate in figura 1 ce prezinta descrierea unei stari initiale date si in figura 2 ce prezinta calea optima gasita.

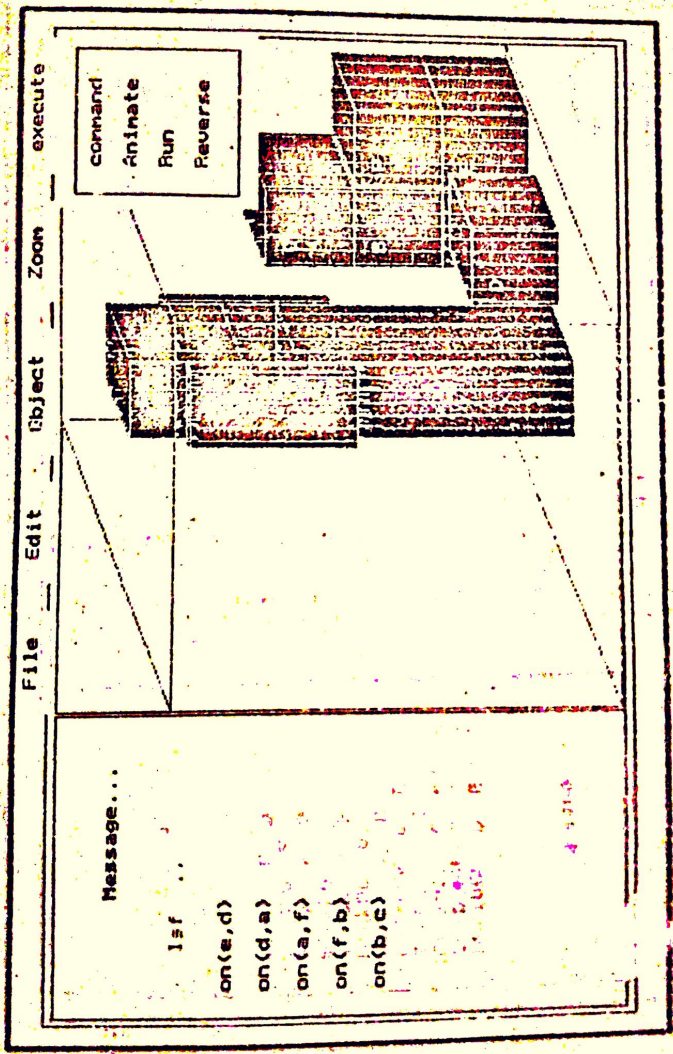


fig. 1

Command
Animate
run
Reverse

Message...

```
UNSTACK C B
PUTDOWN U A
UNSTACK B A
STACK B C
UNSTACK F E
STACK F B
PICKUP A F
STACK A F
PICKUP D A
STACK D A
PICKUP E D
STACK E D
```

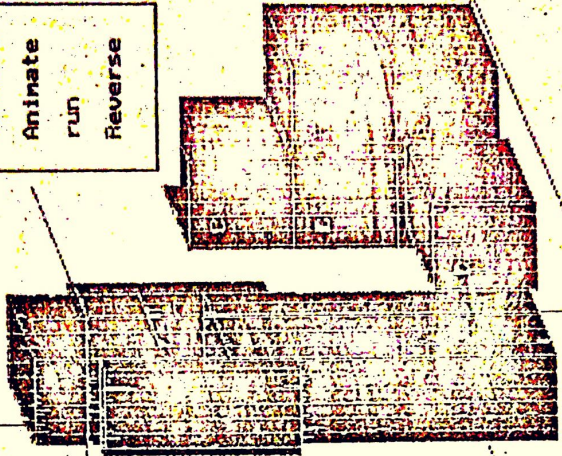


fig. 2

7. BIBLIOGRAFIE

- [1] Niels Nilsson Principles of Artificial Intelligence
Springer Verlag New-York 1982
- [2] Patrick Winston Artificial Intelligence
Addison-Wesley 1977
- [3] * * * Colectia revistelor CAD 1988, 1989
- [4] Dorian Dogaru Elemente de grafica 3-D
Editura Stiintifica si Enciclopedica
Bucuresti 1988

Sorin Ticrea (29 august 1967 CLUJ) a obtinut diploma de inginer in stiinta calculatoarelor la Facultatea de Calculatoare a Institutului Politehnic din Cluj in anul 1991. Domeniul sau de interes major il reprezinta simularea grafica interactiva si inteligenta artificiala.

Calin Rusu (14 ianuarie 1958 DEJ) a obtinut diploma de inginer in specialitatea calculatoare - automatizari la Facultatea de Electrotehnica a Institutului Politehnic din Cluj, in anul 1983. In prezent este asistent la catedra de Actionari Electrice si Roboti Industriali. Preocuparile sale actuale includ Dinamica si Controlul Robotilor, Sisteme Expert de Tiap Real.

ARHITECTURA BIPROCESOR PENTRU APLICĂȚII DE INSTRUMENTAȚIE

Autor: ș.l. ing. Ștefan Poli - I.P.Iași

1. Introducere

Tehnica modernă a măsurărilor este astăzi de neconceput fără utilizarea sistemelor computerizate care să asigure flexibilitate, adaptabilitate la cele mai diverse cerințe, pret de cost relativ scăzut raportat la un grad înalt de performanță. Sistemele de instrumentație au apărut ca o consecință directă a evoluției tehnicii de calcul și mai ales a domeniilor microelectronicii și a microinformaticii. Larga răspândire a microcalculatoarelor (mai ales a celor din familia IBM-PC) a determinat apariția rapidă a interfețelor de măsură, capabile să transforme echipamentul de calcul într-un complex destinat celor mai diverse aplicații: osciloscopice digitale, sisteme pentru analiza și controlul vibrațiilor, analizoare Fourier, analizoare spectrale, sisteme pentru instrumentație analitică în chimie, instrumentație medicală, etc.

Structura unui sistem de instrumentație (fig.1) include în principiu următoarele componente: traductoarele ce prelevează mărimile de măsurat, condiționările de semnal ce realizează preprocesarea analogică a semnalelor (amplificare/atenuare, filtrare, etc.), achizitorul de date care transformă semnalul analogic de intrare într-o mărime numerică, sistemul de calcul care realizează analiza și eventual elaborează deciziile, ieșirile analogice care furnizează semnale prelucrate (eventual comenzi pentru sistemele în buclă închisă) și blocul de postprocesare analogică ce permite interfațarea cu elementele de execuție. Evident unele module pot lipsi în cazul unor configurații mai simple.

Una din problemele principale ridicate de aplicațiile de control industrial (și chiar de cele de măsură) este prelucrarea datelor și elaborarea deciziilor în "timp real". Sistemele de "timp real" sunt caracterizate de posibilitatea de a realiza achiziția datelor și prelucrarea lor într-un interval de timp suficient de scurt în raport cu evoluția procesului controlat (sau măsurat). Pentru orice sistem de achiziție și prelucrare de date sau sistem de control, indiferent de cât de lent ar fi acesta, există aplicații suficient de lente astfel încât sistemul să se comporte ca un sistem de "timp real". Multe aplicații (mai ales cele de măsură) pot fi rezolvate și "off-line" adică achiziția, prelucrarea și eventual elaborarea deciziilor se pot face în etape succesive dar suficient de scurte pentru a păstra un timp de răspuns acceptabil; astfel de sisteme mai sunt cunoscute sub numele de sisteme "pseudo timp real". Creșterea de viteză cerută de multe aplicații se poate obține pe mai multe căi:

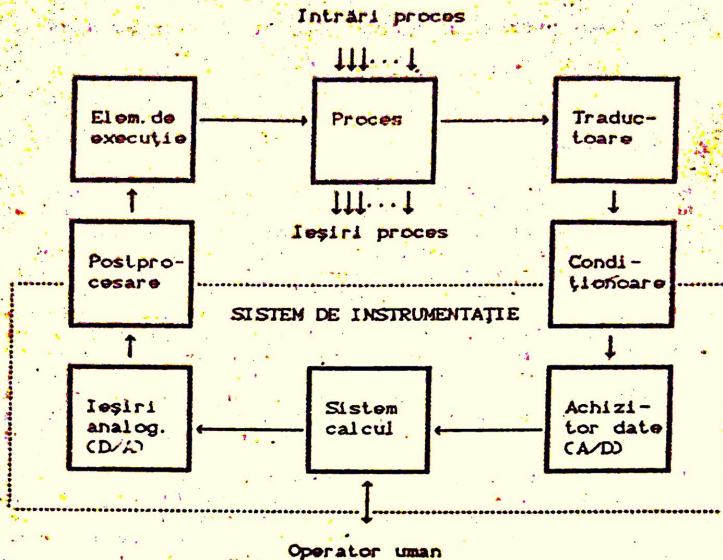


Fig.1 Schema bloc a unui sistem de instrumentație / control

utilizarea unor module de achiziție suficient de rapide, eventual dotate cu unități de prelucrare proprii capabile să asigure și o serie de sarcini locale de stocare și preprocesare a mărimilor culese.

utilizarea unor sisteme de calcul performante (eventual arhitecturi multiprocesor) care să permită prelucrarea rapidă a datelor și elaborare a deciziilor într-un timp cât mai scurt.

2. Prezentarea arhitecturii sistemului

În cele ce urmează este prezentată o arhitectură biprocesor destinată echipării sistemelor de instrumentație și/sau control. Acesta permite realizarea unei stații de lucru relativ complexe care poate îndeplini atât funcții de interfațare cu un proces, cât și prelucrări proprii ale mărimilor achiziționate. Prin conectare cu un microcalculator IBM-PC se poate obține un complex de prelucrare distribuită extrem de flexibil.

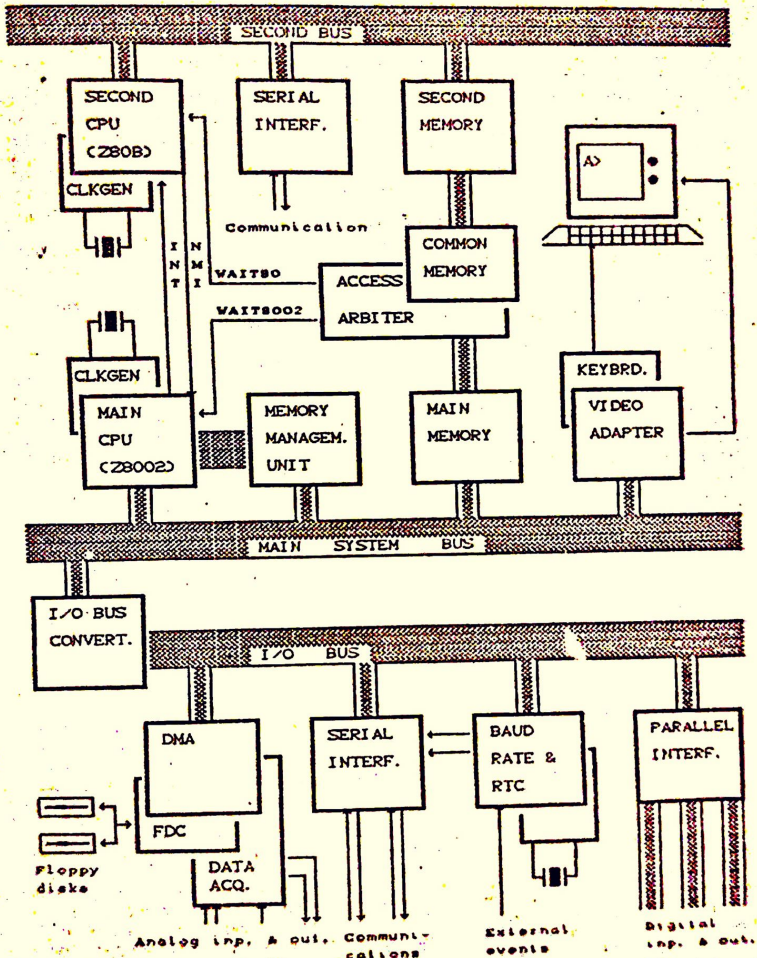


Fig.2 Sistem bi-procesor pentru aplicații de instrumentație

Sistemul este compus din două structuri cu microprocesor interconectate între ele printr-o memorie biport comună (fig.2). Arbitrarea accesului se face pe baza principiului "primul venit - primul servit" cu ajutorul unui automat simplu, care pune în așteptare procesorul solicitant dacă memoria comună a fost alocată celui alt procesor (care o folosește).

Procesoarele folosite sînt microprocesorul Z8002 de 16 biți și microprocesorul Z80B de 8 biți ale firmei ZILOG, dar ele pot fi substituite cu microprocesoare din familia INTEL 8086/8088 fără ca principiul interconectării să se modifice. În acest din urmă caz una din structurile folosite (sau chiar ambele) pot evolua către o arhitectură de microcalculator compatibil IBM-PC. Suportul software se bazează pe sistemul de operare CP/M (pentru Z80B) și un sistem de operare propriu PDOS (pentru Z8002).

Funcțiile realizate de cele două sisteme au fost împărțite astfel încît să poată fi asigurat paralelismul între operațiile de intrare/ieșire (inclusiv cele de achiziție de date) și operațiile de calcul. Astfel primele au fost preluate de sistemul cu Z8002, iar celelalte au fost preluate de sistemul cu Z80B.

a. Sistemul cu Z8002

Utilizarea microprocesorului Z8002 se justifică prin facilitățile oferite de fabricant, caracteristicile sale permițînd asemănarea cu o unitate centrală de minicalculator. O comparație cu alte microprocesoare evidențiază performanțe similare cu MC8800 al firmei MOTOROLA. Ușurința interfațării cu circuite atît din familia ZILOG cit și cu circuite din familia INTEL permit alegerea unei arhitecturi de sistem aliniată la o magistrală standardizată. În cazul de față s-a ales o magistrală MULTIBUS simplificată (dar cu succes se poate alege și PC-BUS/XT, sau PC-BUS/ATD). Explicația alegerii este posibilitatea schimbării procesorului cu unul din familia INTEL 8086/8088 fără un efort considerabil de reproiectare.

Memoria principală este utilizată printr-un dispozitiv de paginare cu pagini de 4kB, care permite adresarea unui spațiu fizic de 16 MB cu numai 64 kB de spațiu logic disponibil procesorului principal. Acesta permite o serie de facilități care cresc considerabil performanțele sistemului:

- posibilitatea stocării datelor achiziționate direct în memorie fără a fi necesară "virtualizarea" prin intermediul discului magnetic (creștere de viteză);
- dezvoltarea de aplicații multitasking fără evacuarea taskurilor suspendate în discul magnetic (de asemenea creștere de viteză);
- utilizarea ca RAM disc în aplicații de mare viteză care reclamă un mijloc de gestionare a fișierelor.

Subsistemul de intrare/ieșiri a fost conceput într-o manieră modernă utilizînd o magistrală suplimentară de 8 biți (I/O BUS) destinată doar transferurilor cu periferia. Au fost prevăzute toate tehnicile de transfer (bucă programată, pe întreruperi și prin acces direct la memorie) astfel încît să se poată obține un maximum de adaptabilitate la cele mai diverse cerințe ale perifericelor și aplicațiilor. Astfel dispozitivele lente

(interfețe seriale, achiziitoare lente, interfețe paralele orientate pe octet) sînt cuplate utilizînd legarea prin buclă programată și pe întreruperi, iar dispozitivele rapide (disc magnetic, achiziitoare de viteză) sînt cuplate utilizînd legarea prin DMA și întrerupere la sfîrșit de transfer.

Sistemul include și controlul întreruperilor vectorizate pe 8 nivele, extensibil la 16.

Se poate observa că funcțiile destinate acestui sistem îl transformă practic într-un procesor de intrare/ieșire și achiziție de semnal pentru celălalt sistem cu care este interconectat. În afara acestor funcții el mai poate prelua și funcții de tip "coprocesor" în cazul în care celălalt procesor este mai puțin performant.

1. Sistemul cu Z80B

Destinat în principal calculului și elaborării deciziilor, acest sistem a fost redus la un minimum de resurse (practic fără resurse de intrare/ieșire). Utilizarea microprocesorului Z80B se justifică doar prin simplitatea implementării. Într-o variantă mai performantă se poate înlocui acest sistem direct cu o arhitectură de tip IBM-PC. Din considerente de viteză s-a ales un procesor dintr-o clasă superioară (6 MHz) care să asigure totuși un minimum de performanțe de calcul.

Memoria principală conține 60 kB RAM și 2 kB ROM, restul de 2 kB fiind destinați să acopere spațiul memoriei comune. Pentru ușurința depanării a fost prevăzută o interfață de comunicații seriale RS232 pentru lucrul în mod local.

c. Interconectarea prin memorie biport

Cele două procesoare pot partaja o zonă de memorie comună de 2kB într-un mod perfect transparent pentru utilizator. Arbitrarea accesului se face după principiul "primul venit - primul servit": eliminarea situațiilor conflictuale se face prin punerea în așteptare a procesorului care face o cerere și găsește resursa (memoria comună) ocupată de celălalt procesor (fig.3 și fig.4).

Memoria biport este organizată în funcție de cerințele procesorului cu magistrala mai mare (Z8000) care impune divizarea în bancuri HIGH și LOW. La nivelul celuialt procesor accesul fiind numai pe octet acesta poate fi dirijat spre partea HIGH sau LOW doar pe baza bitului de adresă A₀ (adresă pară/impare).

Sistemul cu Z8002 "vede" această zonă comună într-o pagină de memorie fizică între adresele FFO:800..FF0:FFF. Ori de cîte ori este necesar un acces în această zonă, se comută prin MMU o pagină logică nefolosită înscriind în registrul de pagină aferent valoarea OFF0H.

Sistemul cu Z80 (coprocesorul) "vede" la rîndul său zona comună între adresele FB00..FFFF, după memoria EPROM (care începe la F000). Zona este în permanentă accesibilă procesorului.

Pentru evitarea conflictelor "software" la comunicația între cele două procesoare a fost prevăzut un mecanism de semnalizare, implementat pe întreruperi. Astfel sistemul cu Z802 utilizează NMI pentru cererea de întrerupere adresată de coprocesorul cu Z80, iar acesta din urmă utilizează INT (codul IM 2) pentru cererea de întrerupere adresată de primul.

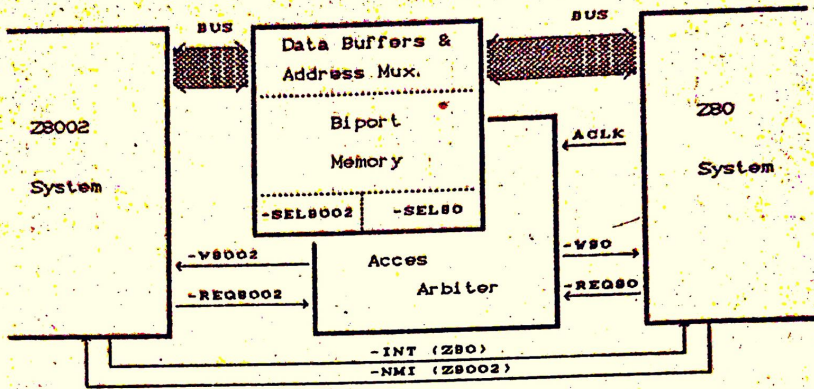


Fig.3 Interconectarea prin memorie biport

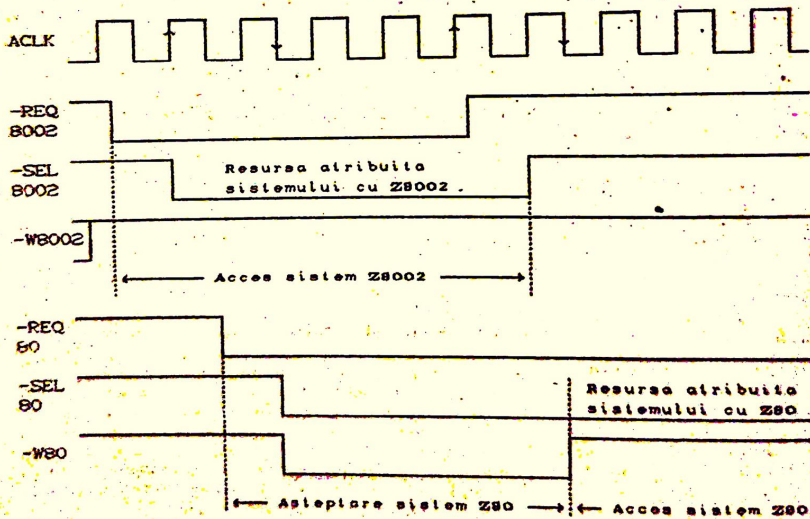


Fig.4 Arbitrarea memoriei comune

Automatul de arbitrare elimină situațiile conflictuale când cele două procesoare cer accesul simultan la memoria comună. O soluție simplă utilizează coasul de arbitrare (ACLIO) asociind frontul pozitiv cererilor sosește de la Z8002 și frontul negativ cererilor sosește de la Z8001. În acest fel alegerea procesorului care va face acces în cazul cererilor simultane, depinde de primul front activ al coasului de arbitrare (dacă frontul pozitiv sosește primul atunci se va atribui resursa procesorului 8002). O descriere posibilă a schemei de arbitrare este dată mai jos.

Secvența de arbitrare a cererilor pentru procesorul Z8002:

```
...
sel8002=w80=0;
if(req8002)
{
    sel8002=1;
    if(req80)
        w80=1;           /* Z80 trece in asteptare */
    else
        w80=0;           /* eliberează accesul Z80 */
    while(w8002);        /* așteaptă accesul Z8002 */
    access_8002();       /* are loc accesul Z8002 */
    sel8002=0;
}
...
```

Secvența de arbitrare a cererilor pentru procesorul Z80:

```
...
sel80=w8002=0;
if(req80)
{
    sel80=1;
    if(req8002)
        w8002=1;       /* Z8002 trece in asteptare */
    else
        w8002=0;       /* eliberează accesul Z8002 */
    while(w80);         /* așteaptă accesul Z80 */
    access_80();        /* are loc accesul Z80 */
    sel80=0;
}
...
```

3. Blocul de interfețe de instrumentație

Subsistemul de achiziție de date analogice permite o gamă largă de utilizări în aplicații pentru care sînt suficiente precizii de 12 biți și viteze de pînă la 50000 de eșantioane pe secundă. Acesta cuprinde 16 canale de intrare multiplexate și două canale de ieșire independente.

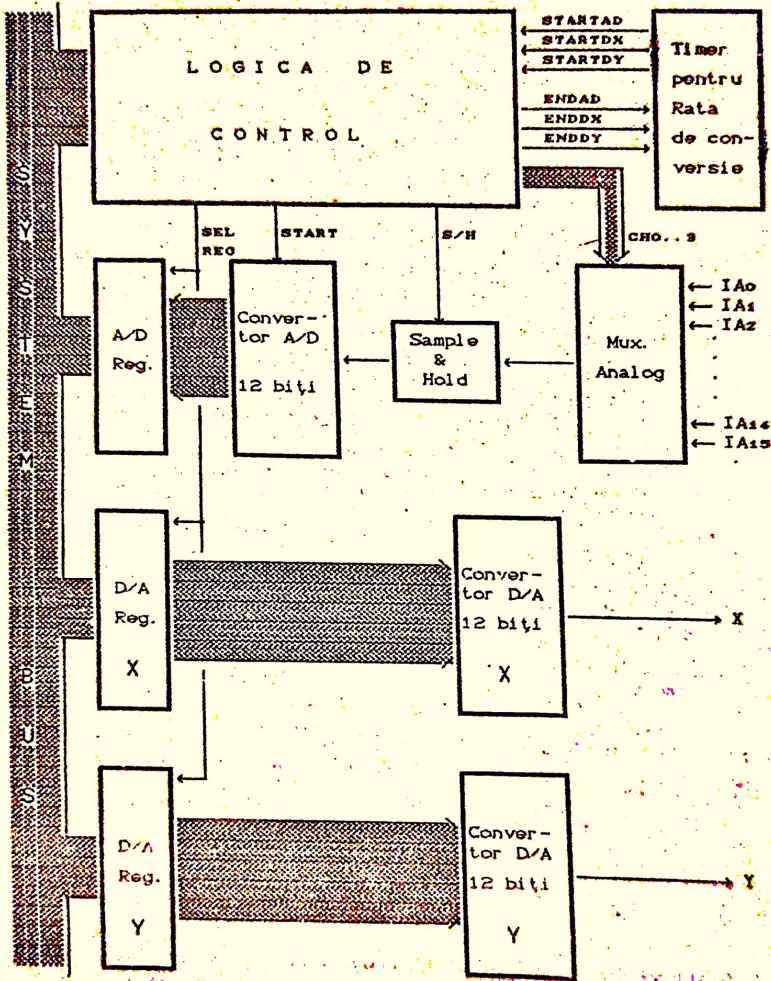


Fig. 5 Subsistemul de achiziție de date analogice

Flexibilitatea subsistemului este asigurată prin alegerea unor tehnici de interfațare adecvate. Astfel toate modulele de transfer (bucia programată, pe întreruperi și DMA) sînt posibile, ceea ce permite o bună adaptare la cerințele aplicației. Dintre caracteristicile acestui modul se amintesc:

Intrări analogice:

rezoluție: 12 biți $\pm 1/2$ LSB
 rată de eșantionare: pînă la 50kHz
 gamă de tensiuni de intrare: $\pm 5V$
 curent de polarizare la intrare: $1\mu A$
 număr de canale de intrare: 16 asimetrice multiplexate
 cod: bipolar complement față de 2 cu ext. de semn la 16 biți

Ieșiri analogice:

rezoluție: 12 biți $\pm 1/2$ LSB
 gamă de tensiuni de ieșire: $\pm 5V$
 rată de conversii: pînă la 50kHz
 impedanță de ieșire: $< 100\Omega$
 cod: bipolar complement față de 2 cu ext. de semn la 16 biți

Pentru a obține o rată de eșantionare precis controlată sistemul posedă un timer programabil pilotat de un oscilator cu cuarț; modificarea ratei se poate face cu o rezoluție de $1\mu s$ în gama $20\mu s..60ms$.

4. Sisteme de achiziție și prelucrare de date analogice distribuite

Realizarea unor configurații de instrumentație complexe ridică probleme de interconectare deosebite. Se cunosc două soluții cu largă răspindire: conectarea directă (internă) pe magistrala calculatorului și conectarea externă printr-o interfață de comunicație standardizată (RS-232, RS-422, IEEE-488).

Conectarea externă (fig.6) prezintă următoarele avantaje: poate fi configurat un sistem oricît de complex, subsistemul de achiziție poate fi plasat la distanță (cît mai aproape de locul de prelevare a mărimilor de măsurat), subsistemul de achiziție poate fi interfațat practic cu orice tip de calculator.

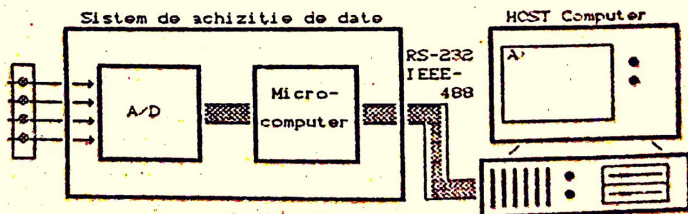


Fig.6 Conectarea externă a unui sistem de achiziție de date

Conectarea internă (fig.7) prezintă și ea o serie de avantaje: viteză mare datorită conectării sistemului direct pe magistrala calculatorului gazdă, preț de cost redus, gabarit mic.

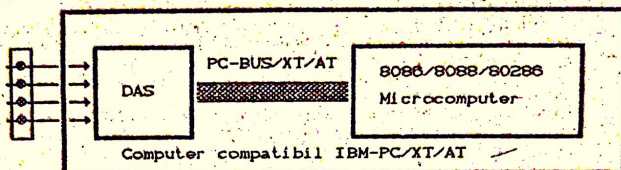


Fig.7 Conectarea internă a unui sistem de achiziție de date

Sistemul de instrumentație prezentat poate fi utilizat atât de sine stătător cit și în configurații distribuite. În acest din urmă caz el poate fi conectat pe unul din canalele de comunicație RS-232 pe care le posedă. Drept "HOST Computer" se poate folosi cu succes orice microcalculator compatibil IBM-PC.

5. Concluzii

Conceput pentru aplicații de instrumentație de uz general, sistemul prezentat a fost realizat pe parcursul a doi ani de studiu a arhitecturilor cu microprocesor și a sistemelor de achiziție de date analogice. Util atât în laborator cit și în aplicații industriale, el poate fi reconfigurat după cerințe.

O serie de îmbunătățiri hardware prevăd echiparea sa cu un achizitor suplimentar pentru aplicații lente, dar care reclamă rezoluții ridicate (16 biți). Acest achizitor, din gama celor cu integrare, va permite și utilizarea în condiții de zgomot mai severe fără complicații software deosebite.

Concepția generală permite și utilizarea în aplicații de control în buclă închisă. În acest scop se încearcă în prezent dezvoltarea unui executiv de timp real care să asigure un suport multitasking eficient.

6. Bibliografie

1. BURR-BROWN The Handbook of Personal Computer Instrumentation - 1989
2. BURR-BROWN Integrated Circuits Data Book vol. 33 - 1990
3. ANALOG DEVICES Data Conversion Products Databook - 1988
4. ANALOG DEVICES Linear Products Databook - 1988
5. IBM IBM-PC/XT/AT Technical Reference - 1986
6. INTEL Microprocessors Handbook - 1986
7. ZILOG Z8000 Technical Manual - 1983
8. ZILOG Microprocessors Components Databook - 1983

METODE DE SPECIFICARE ȘI VERIFICARE A PROTOCOALELOR ÎN REțeleLE DE CALCULATOARE

K. PUSZTAI, V. DADARLAT - Institutul Politehnic Cluj

Rezumat

Scopul acestei lucrări este de a face o comparație între diferitele metode folosite în descrierea și verificarea protocoalelor, folosind ca aplicație un protocol la nivelul transport, protocol clasa 0 pentru o rețea locală.

Aspecte teoretice

O rețea de calculatoare poate fi considerată ca organizată pe nivele, folosind în mod uzual clasificarea I.S.O. pentru sistemele deschise. Structura fiecărui nivel este reflectată prin specificarea protocolului propriu, adică prin descrierea relațiilor și legilor care guvernează comportarea entităților proprii protocolului. Protocoalele reale sînt complicate și se complică odată cu creșterea complexității rețelei, de aceea s-a depus un mare efort pentru găsirea tehnicilor matematice și formale pentru specificarea și verificarea lor.

Printre cerințele esențiale avute în cazul descrierii unui protocol figurează: lipsa ambiguității, posibilitatea analizei lui la diferite nivele de detaliu și o bună structurare.

Căile de specificare a protocoalelor sînt date de limbajele formale sau, în faza incipientă de limbajul natural.

Descrierea prin limbaj natural este în general inadecvată deoarece este lungă, ambiguă iar corectitudinea și completitudinea specificării, greu de verificat.

Metodele formale folosesc modele și pot fi împărțite în cîteva clase, astfel:

... modele tranzitionale, în care comportarea entităților protocolului este descrisă prin mulțimea stărilor și tranzițiilor. Aici pot fi cuprinse rețelele Petri, diagramele de stare, gramaticile, grafuri UCLA. Este o metodă relativ ușor de învățat și apoi analizată (este și foarte folosită de altfel), dar

DIS
UG

pentru protocoalele complexe duce la manipularea unui volum mare de elemente ale mulțimii stărilor și tranzițiilor.

- metode algoritmice, bazate pe limbajele de programare de nivel înalt (Pascal, Pascal concurent, Ada, C++), aici variabilele fiind legate de stările ce descriu comportarea protocolului

- modelele hibride, care combină avantajele celor două metode anterioare; în general ele acționează pe un model tranzițional care reflectă comportarea specifică a protocolului, tranzițiilor fiindu-le asociate module program care operează asupra variabilelor asociate stărilor. Astfel graful stărilor se simplifică deoarece mai multe stări sînt grupate într-o singură stare parametrizată.

În acest sens s-au dezvoltat limbaje de descriere specifice cum ar fi ESTELLE și LOTOS, deja acceptate de CCITT.

În ceea ce privește validarea protocoalelor se folosesc metode diverse, legate de metoda de descriere folosită, cum ar fi: metoda invariantilor, proceduri de reducere asupra expresiilor de transformare a modelului, demonstrarea satisfacerii de entități a diverselor asertiuni, generarea și analiza tuturor stărilor sau a marcărilor într-o rețea Petri etc.

Se încearcă astfel deducerea completitudinii, verificarea ciclicității, absența blocării definitive sau demonstrarea altor proprietăți specifice protocoalelor studiate.

Aspecte ale implementării

Protocolul la nivel transport folosit este deocamdată foarte simplu datorită legăturii fizice aproape perfecte existente, de aceea se poate încadra ca un protocol la nivel transport de clasă 0. Dricum pentru următoare dezvoltări ne vom baza pe existența la nivelul rețea a unui protocol X-25, impunînd considerarea unei rețele orientate pe conexiune.

Rezultă că principalele servicii oferite la nivelul transport sînt mecanismul de gestiune a conexiunii și de transfer date, fără a se trata controlul fluxului și tratarea erorilor.

Descrierea protocolului la nivelul transport s-a făcut în următoarele forme:

- cu ajutorul unei biblioteci de funcții și proceduri Pascal care implementează primitivele de serviciu ale nivelului transport, parte din ele direct accesibile de la nivelul aplicației de tipul file server în acest caz.

- ca o mașină de stări sub forma unei matrici avînd punctele de intrare caracterizate de predicate, acțiuni și stări.

- folosind limbajul ESTELLE (Extended State Transition

Language) - ca un modul continind o colectie de reguli de tranzitii, fiecare regulă aparținind unei stări și specificind care eveniment conditionează tranziția, ce acțiune este îndeplinită și care este starea următoare accesată. Evenimentele și acțiunile sînt module Pascal.

Protocolul elaborat contine cinci primitive de serviciu fiecare cu parametri specifici: connect, send, disconnect - de tip activ, precum și listen, receive - de tip pasiv. Ele sînt direct legate de proceduri din biblioteca nestandard elaborată, dar folositoare aplicației. Pachetele elaborate sînt de tipul 'call request', 'call accepted', 'clear request', 'clear confirm' și 'data'. O conexiune poate fi în una din următoarele șapte stări: idle wait queued ss' hlished send receive disconnect.

Folosind prima metodă de specificare am creat o colecție de proceduri și funcții Pascal ce implementează entitățile protocolului - parte din ele direct accesibile din programul utilizator. Altele, cum ar fi rutinele de întrerupere scrise în limbaj de asamblare sînt specifice și descriu fenomene dinamice (packet arrival, counter clock).

Principala structură de date folosită o constituie descrierea conexiunii și se realizează printr-o înregistrare Pascal cu cîmpuri precum: adresa locală și adresa destinație, starea conexiunii, ceas.

Aplicația fiind de tipul file server se prevăd și subprograme pentru gestiunea nivelelor inferioare, ele realizînd interfata dintre nivelul transport și nivelul inferior.

Folosind a doua cale de descriere, am construit o matrice cu șapte coloane (stările) și douăsprezece linii (evenimentele care schimbă starea: cinci primitive de serviciu, șase tipuri de pachete și un ceas). Fiecare intrare în matrice are trei cîmpuri: predicatul, acțiunea și starea următoare.

Spre exemplu, dacă conexiunea este în stare inactivă dar 'ascultă' pe linie, un pachet de tip 'call request' duce la generarea unui pachet 'call accept' și trecerea în area de stabilire a conexiunii.

Folosind module Pascal, sosirea unui pachet de date de tipul 'call request', dacă variabila 'listen' este poziționată, se poziționează cîmpul 'conn.state' al variabilei conexiune la valoarea 'established' și se generează către nivelul inferior folosind rutina 'to_network_layer' un pachet de tipul 'call accept'.

Prin metoda mașinii de stări finite se setează stările 'idle', 'state2-established', predicatul 'listen' și acțiunea 'call accept'.

Folosind specificația ESTELLE tranziția apare astfel:


```

trans
  from idle
  to established
  when call request
  provided listen
  priority none
  begin
    call accept
  end.

```

Concluzii

Aceste aspecte tratate fac parte dintr-un proiect de cercetare pentru dezvoltarea unui sistem expert dedicat verificării și acceptării protocoalelor de rețea.

Bibliografie:

- Pilat F., Cristea V. - Teleinformatica, Ed. Științifică și Enciclopedică București.
- Stallings W. - Data and Computer Communications, MacMillan Publish. Co
- Nussbaumer P. - Téléinformatique - Ed. Presses Romandes Lausannes
- Tanenbaum A. - Computer Networks, Prentice Hall Intern. Editions
- Goffton P. - Mastering Serrial Communications - Sybex Publish. Comp.

Arhitectura unui sistem multiprocesor de tip cub

Sef. lucrări ing. Dan Grigoras
I.P. Iasi

1. Introducere

Exista numeroase arhitecturi de sisteme multiprocesor, a caror varietate este dictata pe de o parte de adecvanta la o anumita clasa de aplicatii, iar pe de alta de resursele tehnologice disponibile. Se cunoaste ca, cu cit un sistem de calcul este mai paralel, cu atit este mai specializat si, ca un corolar, are performante mai bune.

Totusi sistemele de calcul multiprocesor se pot imparti in doua categorii importante:

- sisteme strins conectate (tightly coupled), cind operatiile de comunicatii, si sincronizare se realizeaza prin memoria comuna (shared memory);
- sisteme slab conectate (loosely coupled), cind aceleasi operatii se executa prin canalele de I/E (message passing).

In primul caz operatiile critice sînt rapide (la timpul de acces al memoriei), dar complexitatea schemelor creste datorita arbitrului de acces la memoria comuna. De aceea, se prefera o organizare ierarhica a spatiului de memorie comuna, tocmai pentru a reduce numarul solicitantilor de acces la un anumit nivel si deci de a simplifica schemele arbitrului. Aceasta solutie creeaza si alte avantaje: posibilitatea de a crea module care pot configura apoi sisteme dupa necesitati; structura ierarhica de tip arbore, hipercub sau clusters (fig. 1) permite o corespondenta mai buna cu numeroase clase de algoritmi paraleli; se pot crea sisteme tolerante la defecte.

La un sistem slab conectat se presupune ca granularitatea paralelismului este mare, deci operatiile de calcul depasesc cu mult pe cele de comunicare sau sincronizare. Astfel, timpul total este putin afectat de interactiunile interprocesoare.

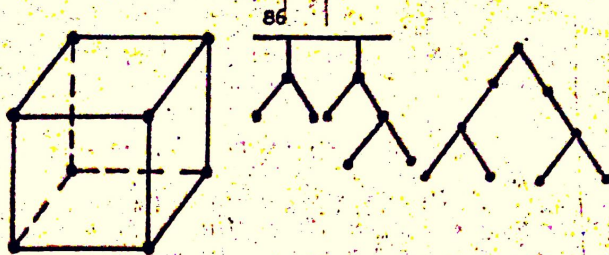


Fig. 1 Topologii de sisteme multiprocesor
• - nod procesor

Mai ales in aceasta situatie, subsistemul software are un rol important in evaluarea volumului de calcul si impartirea lui echilibrata la procesoare.

Abele tipuri de sisteme pot fi de tip stand-alone sau back-end, in acest ultim caz, apare avantajul folosirii mediului de programare a sistemului gazda. Dar si asa interfata intre host si back-end presupune un efort de programare suplimentar. In faza de cercetare se prefera aceasta solutie.

Alegerea nodurilor procesoare depinde de posibilitatile proiectantului, solutiile comerciale utilizand microprocesoarele Intel 80286, 80386, MC 68000 cu coprocesoare din aceeași familie sau WEITEK si din ce in ce mai mult microprocesoare RISC de tip transputer T800.

2. Decizii de proiectare

Avind in vedere observatiile de mai sus am luat urmatoarele decizii:

- am optat pentru o arhitectura de tip strins conectat cu o topologie de cub (fig. 2);

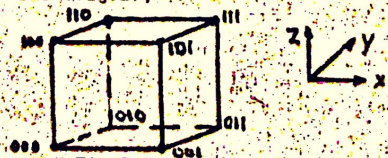


Fig. 2 Topologia de conectare

O astfel de arhitectura permite executarea unei clase foarte mari de algoritmi, în speta cei ce pun în evidență un paralelism geometric sau de tip farming. Soluțiile de tip hipercub folosesc de regula comunicația și sincronizarea prin message passing. Deci, opțiunea noastră, dictată de rațiuni economice este o arhitectura hibrid.

Sistemul MARK II Cosmic Cube sau Caltech Hypercube este modelul de laborator pentru toate arhitecturile de tip hipercub. El a fost proiectat în special pentru rezolvarea problemelor de fizică, ce au un grad înalt de regularitate. Poseda 64 noduri, fiecare nod fiind un microsistem Intel 80286/80387 cu 128 KB, ce comunică cu alți 6 noduri.

Rezultatele publicate pentru executarea a 20 probleme științifice pe cuburi cu dimensiuni între 32 și 128 noduri arată o eficiență medie (viteza de calcul împărțită la numărul de procesoare) de 85%. Pentru anumite probleme aceasta atinge chiar 97%.

Ceea ce diferențiază proiectul nostru de Cosmic Cube este că acesta din urmă nu posedă spațiu de memorie comună; memoria fiecărui nod este adresată numai de procesorul respectiv. Accesul altui procesor la memoria altui nod se face prin transmiterea unui mesaj via o cale de 2-4 biți cu viteza de transfer de 2-4 Mb/s.

- sistemul este considerat un back-end ce se conectează cu un sistem gazdă. Am optat pentru un CUB_2 ca sistem gazdă, datorită opțiunii pentru microprocesorul I-20. Din rațiuni de eficiență, conectarea dintre host și back-end se face prin spațiu de memorie comună. Așa cum se vede în figura 3, calculatorul gazdă este conectat direct cu 2 noduri, prin câte un segment de 168 de biți comună. Alegerea celor 2 noduri, în care este injectată problema de rezolvat, diametral opusă, este dictată de distanța minimă între host și oricare alt nod.

Într-adevăr, datorită topologiei de cub, distanța maximă între host și orice nod al rețelei este 2. În plus, realizarea comunicării între procesoare prin memoria comună și nu pe o cale

seriale, accelereaza acest proces.

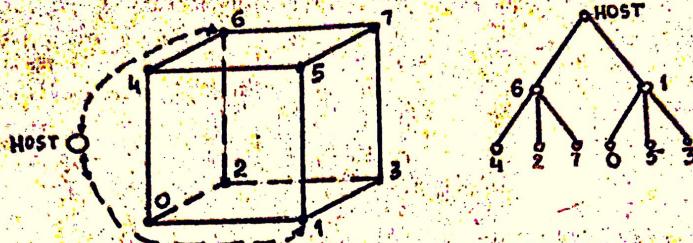


Fig.3 Conectarea host-sistem multiprocesor

3. Comunicarea între procesoare

Pentru comunicarea între oricare două noduri se folosește următorul algoritm: fiind date două noduri, A și B, suma modulo 2 a numerelor lor de adresă conține 1 în dimensiunea unde numerele diferă și, deci, spune după care dimensiune trebuie deplasată data.

Algoritmul va consta în 3 pași (în general k): în pasul 1, data va fi transmisă nodului vecin după dimensiunea 1, dacă bitul 1 al sumei modulo 2 este 1; dacă nu, data nu se deplasează. Deoarece $N = 2^n$, operația de comunicație va consuma cel mult $\log N$ pași.

Totuși, pot să apară conflicte, atunci când la un nod sosesc simultan mesaje ce trebuie transmise pe aceeași direcție. Evident, în acest caz, mesajele trebuie serializate, ceea ce duce la o scădere a performanțelor. O soluție la această problemă constă în calcularea unei matrice generale, compusă din linii ce corespund sumelor modulo 2 dintre toate sursele și toate destinațiile. Pe baza ei, se poate stabili strategia de comunicații, astfel încât să se evite cât mai mult conflictele, fără a scădea prea mult performanțele. Această metodă prezintă dezavantajul unui control centralizat, care trebuie sincronizat pentru toate nodurile.

Să considerăm un exemplu: la momentul t sursa 1 are un mesaj pentru destinația 6, iar sursa 0 pentru destinația 5. Se produce matricea generală:

1	0	1
1	0	1
2	y	x

Se pot evita conflictele, deoarece directia y este folosita numai de sursa 1. Deci principiul metodei consta in cuplarea, in cazul dat, a transmiterii mesajelor dupa o directie fara conflict, pentru o sursa (y pentru 1) si una cu conflict pentru cealalta (x pt. 0), apoi se obtine o noua pereche de directii (x pt. 1 si z pt. 0). In acest moment mesajul transmis de nodul 1 mai efectueaza un pas.

Calculul matricei generale poate reveni procesorului host, care o va efectua in faza de partitionare si distribuire a task-urilor. Apoi, deciziile de orientare a mesajelor vor fi asociate cu momentele impuse prin procesul de calcul al fiecarui task. Astfel, chiar daca datorita naturii MIMD a controlului, este practic imposibil ca toate nodurile sa atinga simultan punctul de unde incepe o operatie de comunicatie ce ar putea genera conflicte, prin metoda propusa, acestea vor fi cu totul eliminate.

Solutia rezolvarii conflictelor la cuburile cu message-passing, este de a agrega mesajele pe directia comuna. Evident, overhead-ul impus de aceasta solutie este important. De aceea se folosesc circuite specializate de tip router.

In cazul nostru, tot procesorul Z-80 este responsabil si pentru controlul operatiei de comunicatii. Prin urmare, in monitorul fiecarui nod se vor afla rutine de tip RECEIVE, SEND. Analiza poate continua cu stabilirea optiunii intre un protocol "store and forward", sau "cut-through" cind fiecare byte al mesajului este transmis la urmatoarul nod imediat ce ajunge. Structura sistemului de calcul proiectat impune adoptarea primului protocol.

4. Concluzii

Sistemul multiprocesor implementat consta in trei microsisteme-nod, realizate cu Z80. Considerentele economice au impus proiectarea minimala a sistemului de calcul. Oricum sistemul realizat permite testarea principalelor concepte legate de

controlul sistemelor de calcul de tip MIMD. De asemenea se pot obtine rezultate concludente privind timpii de executie ai diversilor algoritmi paraleli (numerici sau nenumerici).

Programarea sistemului se realizeaza in limbaj de asamblare, singurele facilitati oferite programatorului fiind cele ale monitorului nodului procesor.

Sistemul realizat face obiectul contractului de cercetare 1049/1989 cu SINTA S.A. si este in faza de testare/dezvoltare.

5. Bibliografie

[1] Almasi, G., Gottlieb, A.: Highly parallel computing, The Benjamin/Cummings Publishing Co., Redwood City 1989.

[2] Hockney, R., Jesshope, C.: Parallel computers, Adam Hilger Ltd, Bristol 1988.

[3] Hwang, K., Briggs, F.: Computer architecture and parallel processing, McGraw Hill, New York 1984.

Soluții de gestiune a resurselor în implementarea structurilor de reglare numerică pe arhitecturi de 8 biți

Octavian Pastrăvanu, Silviu Ibrim

Facultatea de Automatică și Calculatoare

Institutul Politehnic Iași

1. Introducere

Ou toate că arhitecturile pe 8 biți sînt în prezent depășite ca performanțe de calcul prin dezvoltarea unităților centrale de 16 sau 32 biți, larga lor răspîndire și prețurile scăzute le fac încă avantajoase în realizarea structurilor de reglare destinate proceselor lente. Un inconvenient major îl prezintă însă inexistența unui executiv de timp real care să faciliteze proiectarea aplicațiilor ceea ce a determinat ca nici chiar echipamente specializate (ECARM, SPOT) să nu fie utilizate efectiv în reglarea proceselor automatizate. Pe aceste echipamente au fost implementate numai executive de timp real dedicate anumitor aplicații [1], [2]. Singurul nucleu de timp real destinat dezvoltării de aplicații generale, compatibil principal cu aceste arhitecturi este RMX [3], care prezintă dezavantaje majore legate atât de operarea sub SFDX, cît și de lungimea codului executabil ce trebuie înscris în EPROM-uri.

În lucrare sînt analizate posibilitățile de gestionare a resurselor pe baza dispoziția utilizatorului de configurațiile pe 8 biți în implementarea reglatoarelor numerice pentru procese multivariabile lente. Sînt comentate avantajele și dezavantajele fiecărei soluții conturîndu-se totodată caracteristicile unei variante optime. S-a avut în vedere necesitatea desfășurării concurente a următoarelor sarcini:

(S1) - achiziția mărimilor măsurate din proces și elaborarea comenzilor pe baza algoritmului de reglare

(S2) - stabilirea (modificarea) referinței în conformitate cu valorile citite de la un dispozitiv de intrare

(S3) - scrierea (afișarea) valorilor variabilelor considerate de interes în urmărirea procesului.

(S1) este cea mai prioritară și trebuie executată la intervale de timp predeterminate, egale cu perioada de eșantionare, întrerupînd (S2) sau (S3). (S3) are prioritate cea mai scăzută, pentru un dispozitiv I/O comun (de exemplu consola) (S2) întrerupe (S3).

2. Utilizarea aritmeticii în virgulă mobilă în implementarea algoritmulor de reglare

Fie $g_j(z) = \{g_{ij}(z)\}$ ($i=1, p$, $j=1, m$) matricea de transfer a regulatorului discret multivariabil activat de canalele de eroare, ordinul funcțiilor $g_{ij}(z)$ depinzând de complexitatea algoritmului utilizat. Elaborarea propriu-zisă a comenzilor se realizează pe baza ecuațiilor cu diferențe corespunzând funcțiilor $g_{ij}(z)$ iar precizia cerută în efectuarea calculelor impune utilizarea aritmeticii în virgulă flotantă. Algoritmul operând cu unități ingineresti, trebuie realizate conversiile unități CAN - unități ingineresti - unități CNA.

Cel mai avantajos mod de programare al aplicației este apelarea rutinelor adecvate dintr-o bibliotecă de firmă compatibilă CP/M [4] care oferă totodată garanții în exploatare incomparabile cu colecțiile de rutine concepute artizanal. Apelul se poate realiza din limbaj de asamblare sau direct din limbajul de nivel înalt ceea ce constituie o modalitate foarte comodă de utilizare.

Pentru cele trei biblioteci standard caracteristicile sînt:

- a) FORLIB.REL - apelabilă din FORTRAN cu o reprezentare în virgulă flotantă pe patru octeți și durată maximă de execuție a unei înmulțiri în virgulă flotantă de circa 3 ms.
- b) FPREAL.REL - apelabilă din PASCAL MT+, caracteristici asemănătoare cu cea anterioară.
- c) MATH.REL - apelabilă din C AZTEC; toate operațiile se efectuează în dublă precizie baza reprezentării fiind 256. Întrucît rutinele bibliotecii însăși sînt scrise în C, durată unei înmulțiri este de circa 25 ms.

3. Realizarea driverelor pentru circuitele de interfață

Elaborarea comenzii de către microcalculatorul regulator necesită utilizarea unui ceas de timp real care să marcheze momentele de eșantionare. Programarea acestuia se realizează curent în limbaj de asamblare dar se poate face comod și în C.

Driverelor pentru lectura convertoarelor A/N și scrierea convertoarelor N/A se implementează convenabil fie în limbaj de asamblare fie în C.

Observații similare sînt valabile și pentru programarea circuitului de arbitraj a intreruperilor (dacă există în configurație) în vederea rezolvării solicitărilor simultane la resurse [5].

4. Gestiunea datelor în multitasking

Necesitatea rezolvării concurente a sarcinilor (S1), (S2), (S3) impune o disciplină în manipularea datelor specifică rutinelor reentrante. Inexistența

unor astfel de rutine de firmă compatibile CP/M, care să execute calculele aritmetice și operațiile de I/O reprezintă în fond inconvenientul fundamental în exploatarea în aplicații de conducere numerică discretă a arhitecturilor pe 8 biți.

Soluția propusă în lucrare și experimentată în numeroase aplicații se bazează pe următorul mod de gestiune a datelor: taskul A (provenind din modulele scrise atât în limbaj de asamblare cât și în limbaj evoluat) este linkeditat separat de taskul B și taskul C (provenind din module scrise atât în limbaj de asamblare cât și în limbaj evoluat) care fac obiectul unei alte linkeditări. Cele două programe executabile comunică între ele prin variabile din blocuri COMMON (pentru limbaj FORTRAN) sau prin variabile externe (pentru PASCAL sau C). Fiecare zonă de comunicație este ușor de realizat în cazul blocurilor COMMON întrucât alocarea în FORTRAN este statică. În cazul alocării dinamice s-a dovedit necesară o gestiune proprie a stivei deoarece programele executabile rezultate din cele două linkeditări distincte alocă stiva în același punct. Această gestiune este un consumator suplimentar de timp la lucrul în întreruperi deoarece se forțează puncte de intrare diferite printr-o rutină anume concepută. Cele două fișiere conținând codul executabil sînt salvate împreună alcătuiind în exploatare un singur program, eliminînd astfel inconvenientul nereentrantei rutinelor de bibliotecă.

Transferul datelor între cele două executabile este organizat numai pe variabile întregi; întrucît procesarea prin instrucțiuni UC pe dublu cuvînt este neîntreruptibilă, nu se permite alterarea variabilei datorită apariției unei întreruperi în timpul transferului. Se evită astfel procedeul uzual de protejare a secvențelor neîntreruptibile prin dezactivarea și reactivarea sistemului de întreruperi, metoda care foloseită excesiv poate provoca o eșantionare neuniformă în cazul perioadelor de eșantionare mici.

5. Realizarea executabilelor probabile

În aplicațiile industriale curente nu are sens utilizarea mașinilor CP/M, ci se realizează structuri dedicate avînd software-ul integral înregistrat în EPROM-uri.

Dacă este folosit drept limbaj evoluat C-ul în implementarea algoritmului de reglare, linkeditorul LN poate realiza direct forma probabilă prin poziționarea corespunzătoare a secțiunilor de cod și de date; biblioteca standard ROM.LIB asigură funcțiile sistem, iar în rutinele I/O sînt necesare unele intervenții în funcție de tipul dispozitivului.

Dacă se folosește drept limbaj evoluat Pascal MT Plus, în apelul linkeditorului LINKMT se utilizează opțiuni pentru a preciza adresa codului, respectiv a datelor și trebuiesc operate modificări în rutinele I/O.

înlocuind adecvat apelurile în rezidentul CP/M-ului.

Pentru FORTRAN este valabilă aceeași metodologie de lucru folosind linkerul L80.

Este deosebit de important că executabilele rezultate din module scrise în FORTRAN sau Pascal MT Plus sînt semnificativ mai reduse decît cele rezultate din module C.

6. Variante de implementare a unui regulator numeric pentru procese cu două intrări și două ieșiri

Au fost elaborate și testate în vederea unui studiu comparativ două versiuni scrise în limbaj de asamblare și C, respectiv în limbaj de asamblare și FORTRAN, ambele implementînd un același algoritm de reglare pe un echipament tip ECAROM. Experimentele au fost realizate pentru procese bivariabile modelate pe calculator analogic (de ex. [6]) sau pe instalații cu două mîrimi de intrare și două de ieșire (de ex. [7]). Funcțiile de transfer $g_{ij}(z)$, $i=1,2$, $j=1,2$ ale regulatorului implementat în [6] sînt de ordin 2, iar pentru regulatorul utilizat în [7] sînt de ordin 3.

În fig.1 se prezintă organizarea generală a aplicației. Comenzile se calculează în cadrul taskului TK1 lansat în execuție la intervale de timp egale cu perioada de eșantionare T de către întreruperile ceasului de timp real; TK1 include de asemenea și un algoritm anti-windup. TK2 și TK3 permit interfațarea cu operatorul: TK2 preia de la consolă valorile referințelor pentru cele două canale, iar TK3 afișează mîrimea măsurate, erorile și comenzile. TK1 este prioritar în raport cu TK2 și TK3 cu care comunică (în maniera prezentată în secțiunea 5) prin variabile întregi avînd funcționalitatea unor căsuțe poștale CP. Acest mod de organizare asigură sincronizarea, excluderea mutuală și comunicația între taskuri, avînd loc desfășurarea paralelă fie a taskurilor TK1 și TK2, fie a taskurilor TK1 și TK3.

6.1 Variante realizată în limbaj de asamblare și C

Pentru funcțiile de transfer $g_{ij}(z)$, $i=1,2$, $j=1,2$ de ordin 2 dimensiunea finală a programului este de 30 Ko iar perioada de eșantionare minimă cu care s-a operat a fost de 0.8s. Această variantă prezintă avantaje în programare putîndu-se interpola cu ușurință module scrise în C cu module scrise în limbaj de asamblare.

6.2 Variante realizată în limbaj de asamblare și FORTRAN

Pentru funcțiile de transfer $g_{ij}(z)$, $i=1,2$, $j=1,2$ de ordin 2 dimensiunea finală a programului este de 13Ko. S-a putut opera cu o perioadă de eșantionare minimă de 0.1s.

În ce privește programarea, apelurile din limbaj de asamblare în FORTRAN

și diverse presupun un mod mai laborios de transfer al parametrilor prin blocuri comune. Deosebita realizarea versiunilor promabile necesită intervenții în rutinele ITYIN și ITYOT (din biblioteca FORLIB care deservea operațiile I/O) în sensul înlocuirii apelurilor în BIOS prin ecverțe de tratare proprii.

7. Concluzii

Studiul comparativ elaborat pe parcursul secțiunilor precedente a permis să se pună în punct o metodologie de proiectare și implementare a aplicațiilor de conducere numerică directă a proceselor multivariable lonto folosind arhitectura de 8 biți. Această metodologie se bazează pe facilitățile oferite de mașina CP/M în elaborarea și testarea programului înainte realizării versiunilor promabile.

Se poate afirma pe seama tuturor observațiilor anterioare că în ansamblu modul de programare cel mai avantajos al acestor aplicații se realizează în limbaj de asamblare și FORTRAN care conduce la viteza de lucru și imaginea executabililor cele mai favorabile. Cu toate facilitățile oferite de programarea în C sau PASCAL pentru mașinile pe 16 sau 32 biți, pentru configurațiile pe 8 biți implementările cele mai eficiente se bazează pe modulele de calcul aritmetic ale bibliotecii FORLIB.

8. Bibliografie

1. Dumitrache, I. ș.a. "Direct Digital Control of Biosynthesis Processes". Prepr. 7th Int. Conf. on Control Systems and Comp. Science, vol. 1, pp. 76-80, București 1987
2. Munteanu, P.I., Mucă, Gh. "Process Control Application Based on a Dedicated Real Time Executive". Prepr. 7th Int. Conf. on Control Systems and Comp. Science, vol. 1, pp. 126-133, București 1987
3. *** RMK-80 User's Guide, Intel
4. Moraru, Fl., Atodiroaiei, M. "Programarea microcalculatoarelor în sistemul de operare CP/M", Ed. Științifică și enciclopedică 1989
5. Petrescu, A. ș.a. "Microcalculatoarele Felix M18, M18B, M118", Ed. Tehnică București 1984
6. Pastravanu, O., Ona, Al., Schönberger, Fr. "On-line Tuning of Digital Controllers for Multivariable Plants with Unknown Parameters", Proc. ICATB '91, Craiova 1991 (în curs de apariție)
7. Lazăr, C., Schönberger, Fr., Ona, Al., Pastravanu, O., Trelea, C. "Conducerea numerică directă a unui cuplor de tratament termic", al III-lea Simpozion Național - Structuri, algoritmi și echipamente de conducere a proceselor industriale, Iași 1991

DIS

210



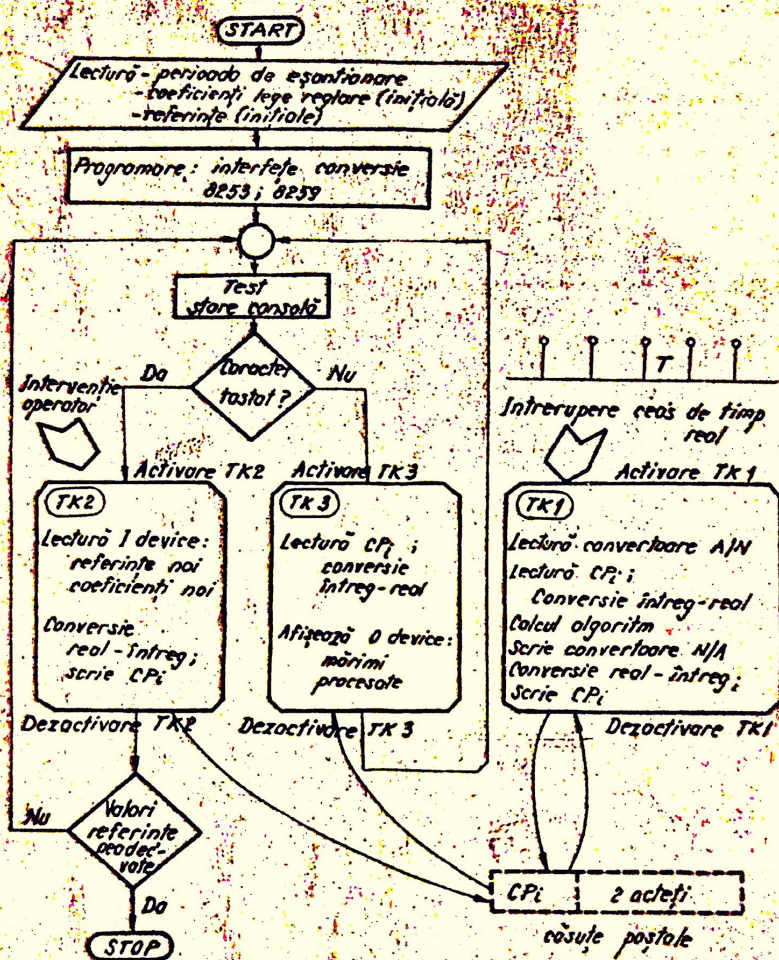


Fig. 1. Organizarea generală a aplicației de timp real

DIS

PLANIFICAREA TASK-URILOR IN SISTEMLILE DE CONDUCERE A PROCESELOR

autorii: as.ing. M. Litoiu, I. F. Bucuresti
Fac. Automatica

ing. M. Litoiu, I. N. M. T. Bucuresti

REZUMAT

Se considera un sistem in timp real cu un singur procesor. In care planificarea task-urilor se face dupa prioritati stabilite in faza de proiectare. Sunt prezentate cteva rezultate ale autorilor, rezultate care confera instrumente simple si puternice pentru alocarea prioritatilor astfel incit, in urma aplicarii acestora, programatorul sa aiba siguranta obtinerii unui sistem inalt performant.

1. CONSIDERATII GENERALE

Sistemele pentru conducerea proceselor desfasoara o activitate ce se repeta periodic pe un interval $[t, t+M]$, unde $M \gg 0$.

```
REPEAT  
  PROGRAM_DE_CONDUCERE;  
UNTIL <conditie>.
```

Deoarece PROGRAM_DE_CONDUCERE e alcătuit din task-uri, proprietatea de periodicitate a acestuia se transmite asupra partilor componente. Deci, intr-un sistem de conducere (ca si in alte sisteme in timp real) task-urile sînt periodice:

```
REPEAT  
  task 1, task 2, ..., task n;  
UNTIL <conditie>.
```

unde task $i, i=1, n$ sînt task-uri independente sau care se sincronizeaza in anumite puncte sau momente de timp. Dar, sistemul, fiind in timp real, se impune si alta cerinta: task-urile trebuie terminate la fiecare reluare, intr-un interval de timp bine determinat (deadline). Acest interval nu poate depasi perioada de reluare. Implementarea task-urilor se face prin intermediul unor sisteme de operare in timp real. Partea din sistemul de operare care efectueaza lansarea in executie a task-urilor se numeste planificator (scheduler). In foarte multe sisteme de operare in timp real planificarea task-urilor se face in functie de prioritatile atribuite task-urilor in faza de implementare a acestora. In capitolul urmat vor fi prezentate un model matematic si o teorema de optima pentru acest caz.

2. MODEL-MATEMATIC

Se poate defini modelul ce va fi analizat ca un sistem:

(1) $S = \{C(t), E(t), I(t), O(t), T(t)\}$, unde

ET

$T_i, i=1, n$ este task-ul cu numărul i conținut dintr-o secvență de program ce se cere executată periodic, a.i.:

a) Executia cu numărul k se cere la momentul

(2) $toi + (k-1) \cdot li$; $k=1, 2, \dots, li$ este perioada lui T_i ;

toi desemnează momentul primei cereri;

b) Executia task-ului T_i cere li unitati de timp.

c) Executia cu numărul k trebuie terminata nu mai tarziu de $toi + (k-1) \cdot li + di$, $0 \leq di \leq li$

d) Executia task-urilor se face pe un singur procesor.

e) Task-urile sînt independente, adica nu exista nici o relatie de ordine între ele.

f) Task-urile au asignate prioritati, astfel incit, daca sînt mai multe cereri de executie simultan, primul se executa task-ul cu prioritatea cea mai mare. Executia acestuia se face cu suspendarea task-ului curent (preemptibilitate), sau dupa terminarea task-ului curent (nepreemptabilitate).

Modelul prezentat corespunde unui numar important de situatii din practica sistemelor in timp real. In particular, in cazul sistemelor de conducere a proceselor, cerinta e) poate aparea ca restrictiva, deoarece in acest caz exista si relatii de ordine între task-uri (intii se face achizitia, apoi prelucrarea, apoi se elaboreaza comanda, etc.). Aceasta situatie poate fi eliminata grupind task-urile dependente intr-unul singur. Modelul prezentat acopera si alt caz intilnit in practica programarii sistemelor si anume situatia in care programatorul foloseste doar intreruperile sistemului pentru a acoperi necesitatile de timp real. In aceasta situatie intreruperile joaca rolul task-urilor din modelul prezentat. Indiferent daca se lucreaza cu task-uri sau doar cu intreruperi, perioadele li ca si duratele i pot fi determinate pentru fiecare task. (Pentru task-urile care nu au o perioada naturala sau durata de executie constanta se considera cazul cel mai dezavantajos: li minim si li maxim).

Definitia 1. Un algoritm A accepta sistemul S daca planifica executia task-urilor astfel incit sa fie respectate toate intervalele di , oricare ar fi momentele initiale $toi, i=1, n$.

Cerinta ca sistemul sa fie acceptat oricare ar fi conditiile initiale nu este exagerata si corespunde situatiei reale din practica. Modelul (1) ales pentru analiza descrie sistemul real in situatiile cele mai dezavantajoase (cind li sînt minime si toi maxime), momentele cind se realizeaza toate aceste situatii neputind fi cunoscute aprioric.

Definitia 2. Fiind data o alocare oarecare a prioritatilor in care task-urile T_m si T_p au alocate prioritatile $k+1$ si respectiv k si $Ip > Im$, se numeste permutare de tip P schimbarea prioritatilor task-urilor T_m si T_p astfel incit T_m sa aiba prioritatea k si T_p prioritatea $k+1$.

In continuare consideram, pentru simplificare, $di=li$ (un task trebuie executat pina la urmatoarea cerere) si cazul preemptibil.

LEMMA. Daca o alocare oarecare a prioritatilor accepta (3) sistemul, atunci alocarea obtinuta printr-o per-

mutare P accepta, deasemenea, sistemul.

Demonstratie.

Atunci cind planificarea task-urilor se face dupa prioritati si task-urile sint preemtabile, executia task-urilor de la niveluri mai mari de nivelul $k+1$ nu depinde de prezenta task-urilor cu prioritati mai mici decit $k+1$ (prioritatea $k+1$ este mai mare decit prioritatea k). Task-urile de la nivelul $k+1$ in jos se executa in intervalele ramase 'neocupate' de task-urile cu prioritati mai mari. Deasemenea pozitia pe axa timpului a intervalelor 'neocupate' ramine aceeaasi, chiar daca se realizeaza permutari (de tip P) intre task-urile aflate mai sus de nivelul de prioritate $k+1$ (pastrind aceleasi conditii initiale). Task-ul ce trece de la o prioritate mai mica la una mai mare (printr-o permutare P) vafi 'acceptat', deoarece are la dispozitie intervale neocupate mai mari sau egale decit cele avute inainte de permutare (cind era 'acceptat'). Aceste lucruri indica faptul ca printr-o permutare P (intr-o alocare ce accepta sistemul) singurul task care poate fi "defectat" (executia lui nu se poate efectua in perioada proprie) este cel ce trece de la o prioritate mai mare la una mai mica. Fie $T_p(k+1)$ si $T_m(k)$ task-urile asupra carora se aplica permutarea P. Aceste task-uri au perioadele I_p si I_m aflate in relatia $I_p > I_m$ si prioritatile $k+1$, respectiv k .

Presupunem, prin absurd ca in urma permutarii P, executia lui T_p nu e satisfacuta in intervalul $[t', t'+I_p]$, capetele acestui interval constituind doua cereri consecutive iar $t'+I_p$ (conform ipotezei $d_i=1$) reprezinta momentul pina cind poate fi satisfacuta cererea din momentul t' . De-a lungul intervalului $[t', t'+I_p]$ task-urile T_p si T_m au la dispozitie acelasi 'spatiu neocupat', atit inainte cit si dupa efectuarea permutarii (pastrind aceleasi conditii initiale). Prezenta un exemplu in figura 1.



fig.1

↑ cereri de executie
 ▨ spatiu neocupat

$T_i(j)$ task-ul T_i are prioritatea j .

Task-ul $T_p(k)$ poate fi defectat doar de $T_m(k+1)$ care ii 'consuma' (datorita trecerii pe un nivel de prioritate mai mare) din 'spatiul neocupat' avut la dispozitie inainte de permutarea P. Sint posibile doua alternative:

a) Pe intervalul $[t, t+tp]$ sînt executate porțiuni din $Tm(k+1)$, corespunzînd la cel puțin două cereri ale acestuia. Fie t' timpul pînă la care trebuie satisfăcută execuția ultimei porțiuni din $Tm(t')$ corespunde cu o cerere de execuție a lui Tp care nu trebuie terminată în intervalul $[t, t+tp]$. Atunci:

$$(4) \quad t' - t - D - (r + r') \cdot \tau_m < \tau_p$$

D reprezintă timpul ocupat de task-urile cu priorități mai mari decît $k+1$;

$r \in \mathbb{N}$ și reprezintă numărul de perioade ale task-ului Tm , cuprinse în întregime în intervalul $[t, t']$;

$r' \in \mathbb{R}$ și reprezintă fracțiunea din m rămasă neexecutată de la o cerere anterioară momentului t' , dar cu timpul limită în intervalul $[t, t']$.

Dar, în aceleași condiții initiale, înainte de permutare, cu $tp(k+1)$ și $Tm(k)$

$$(5) \quad (t' - t) - \tau_p - D > (r + q) \cdot \tau_m$$

Deoarece $q > r'$ (dacă cererea lui $Tm(k+1)$ nu a fost satisfăcută înainte de t' , cu atît mai puțin va fi tratată pentru $Tm(k)$), relația (4) contrazice relația (5). Înseamnă că în condiția a) presupunerea că permutarea P nu conservă acceptarea este falsă.

b) Pe intervalul $[t, t']$ sînt executate porțiuni corespunzînd unei singure perioade din task-ul Tm , în acest interval fiind prezenta o cerere, dar timpul limită corespunzător este $t' > t$.

$$(6) \quad tp - r \cdot \tau_m - D < \tau_p$$

$r < 1$ și reprezintă porțiunea din m ce se desfășoară în $[t, t+tp]$.

Dar, înainte de permutare, deci cu $Tm(k)$ și $tp(k+1)$, în tom ales astfel încît $t' = t + tp$ sistemul este acceptat oricare ar fi condițiile initiale, este îndeplinit condiția:

$$(7) \quad tp - \tau_p - D > \tau_m = tp - \tau_m - D > \tau_p$$

Deoarece (6) contrazice (7) înseamnă că și în acest caz presupunerea că în urma permutării P proprietatea de acceptare nu este îndeplinită, este falsă.

TEOREMA. Alocarea priorităților în ordine nedescrescătoare a perioadelor li este optimă (Optimal are următorul sens: dacă o alocare parecine accepta sistemul atunci și alocarea în ordinea specificată mai sus accepta sistemul).

Demonstrație

Conform acestei teoreme, task-ul cu perioadă minimă va avea prioritatea maximă. Fiind dată o asignare oarecare a permutărilor, asignare ce accepta sistemul, se aplică succesiv permutări de tip P pînă se obține alocarea în ordine nedescrescătoare a perioadelor li . Conform lemei (3), permutările de tip P conservă acceptarea sistemului.

3. CONCLUZII

Aa demonstrat ca alocarea priorităților în ordine nedescrescătoare este optimă. Dar nu este suficient. Pentru a verifica acceptarea sistemului este propus un algoritm pentru task-uri reprezentabile, în condiții inițiale $t_01=t_02=...=t_0n=0$.

Se notează:

Vectorul perioadelor T ; $T=(t_1, t_2, ..., t_n)$

Vectorul duratelor Z ; $Z=(z_1, ..., z_p)$

Vectorul de stare al task-urilor $x=(x_1, x_2, ..., x_n)$

$x_i=1$: procesul i este activ (asteapta executia)

$x_i=0$: procesul i este inactiv (a fost executat si nu e cerut inca executia).

k_n, k_n variabile de lucru.

Cu aceste notatii se construiește algoritmul care va răspunde prin "da" sau "nu" la întrebarea: alocarea în ordine nedescrescătoare a priorităților acceptă sistemul?

T
1. $k_n=0$; $x=(1, 1, ..., 1)$; accept='yes'; $t=0$
repeat

$$2. j = \sum_{i=1}^n x_i + \prod_{k=1}^n x_k$$

(9)

3. $t=t+T_j$
pentru $i=1, n$

$$4. k_i = \left\lfloor \frac{t}{T_i} \right\rfloor$$

if $k_i > k_{i-1}$ then

if $x_i=1$ then

accept='no'

exit

else

$k_{i-1}=k_i$

$x_i=1$

until $j=1$

write accept.

Pentru simplificarea notațiilor s-a presupus ca task-ul T_n are prioritatea n (cea mai mare), task-ul T_{n-1} are prioritatea $n-1$, etc. Algoritmul se termină atunci când la unul din task-uri s-a depășit o "depășire" (a apărut o cerere de tratare fără ca cea anterioară să fi fost tratată), sau atunci când task-ul 1 a fost executat, ieșind astfel din situația cea mai critică (când e ultimul dintr-o coadă de lungime maximă).

Teorema (8) și algoritmul (9) oferă două instrumente puternice pentru alocarea priorităților task-urilor și/sau întreruperilor. Recomandările diverselor autorități de alocare priorităților în ordinea importanței task-urilor se dovedesc a fi lipsite de conținut, căci nu importanța este cea care determină prioritatea, ci perioada T_i unui task foarte important i se poate alocă

prioritatea minima, cu conditia ca el sa aiba perioada T_i cea mai mare. Desigur ca de la bun inceput am considerat ca tratam cazul unui sistem care nu are voie sa lase evenimente netratate.

Importanta teoremei (8) este foarte mare, caci ea simplifica obtinerea unui raspuns la intrebarea: "Exista o alocare a prioritatiilor care sa accepte sistemul?". Prin teorema (8) raspunsul la intrebare se da prin studierea alocarii in ordine nedescrescatoare a perioadelor T_i . Fara aceasta teorema, realizarea unui program care sa dea un raspuns acestor intrebări prin evaluarea tuturor asignarii posibile, ar implica foarte mult timp de calcul (daca $n=10$ ar trebui studiate $10!$ alocari). Deasemenea, in cazul utilizarii unui algoritm cu ramificare si marginit (branch and bound algoritm), spatiul de memorie ocupat precum si complexitatea algoritmului fac practic imposibila utilizarea acestuia.

BIBLIOGRAFIE

Coffman, E.G. JR., Computer and Job-Shop Scheduling Theory, John Wiley & Sons, 1976.

prioritatea minimă, cu condiția ca el să aibă perioadă T_i cea mai mare. Deși s-a văzut că de la bun început am considerat ca tratând cazul unui sistem care nu are voie să lase evenimente netratate.

Importanța teoremei (8) este foarte mare, căci ea simplifică oferirea unui răspuns la întrebarea: "Există o alocare a priorităților care să accepte sistemul?". Prin teorema (8) răspunsul la întrebare se da prin studierea alocării în ordine nedescrescătoare a perioadelor T_i . Fără această teoremă, realizarea unui program care să dea un răspuns acestor întrebări prin evaluarea fiecărei asignări posibile, ar implica foarte mult timp de calcul (dacă $n=10$ ar trebui studiate $10!$ alocări). Deasemenea, în cazul utilizării unui algoritm cu ramificare și margină (branch and bound algoritm), spațiul de memorie ocupat precum și complexitatea algoritmului fac practic imposibilă utilizarea acestuia.

BIBLIOGRAFIE

Coffman E.G. JR., Computer and Job-Shop Scheduling Theory, John Wiley & Sons, 1976.

ALEGEREA ELEMENTELOR DE CALCUL ÎN SISTEMLOR DE CONDUCERE A PROCESELOR

as. ing. Marin Iltioiu - I.P. București
Facultatea Automatică

Rezumat

În lucrare este prezentată o metodologie de alegere a sistemului de calcul înglobat într-o instalație de automatizare. Criteriile ce stau la baza acestei alegeri pot fi ponderate astfel încât alegerea să fie optimă, satisfăcând toate cerințele impuse de procesul tehnologic și de beneficiar. Deoarece în faza de proiectare o mare parte din cerințe au un pronunțat caracter vag, pentru alegerea optimă a sistemului de calcul se folosește o metoda bazată pe mulțimi vagi.

1. Metodologii de proiectare

Diversitatea instalațiilor de automatizare face tot mai dificilă sarcina proiectantului de a alege instalația optimă pentru un proces dat. În automatică, ca și în alte domenii, proiectantul are la dispoziție o gamă foarte largă de mijloace dar foarte puține elemente de sinteză. De aceea apare necesitatea elaborării unei metodologii de proiectare a instalațiilor de automatizare, metodologie care să conțină elemente ce-i permit proiectantului o evaluare rapidă și corectă a performanțelor diferitelor variante de elemente de calcul (calculatoare, automate programabile, elemente distribuite, etc). Instalația tehnologică se presupune dotată cu traductori și elemente de execuție.

În figura 1 și în figura 2 sunt prezentate două metodologii propuse pentru generarea unei soluții de automatizare optime. Prima metodologie presupune generarea tuturor soluțiilor posibile pe baza caietului de sarcini și a bazei de date de care dispune proiectantul și apoi aplicarea unui algoritmul de selecție a soluției optime. Se obține un optim global ceea ce este foarte convenabil. Dar, la generarea tuturor soluțiilor posibile, chiar în prezenta unei baze sărace de elemente de calcul, este adesea foarte greu. Riscul acestei metodologii este că va genera optim global doar peste un număr limitat de soluții.

A doua metodologie este destinată să elimine dezavantajul de care am vorbit la prima și face apel la proiectarea structurată a instalațiilor de automatizare.

Proiectarea structurată a instalațiilor de automatizare presupune, în principiu, două etape importante:
- proiectarea arhitecturii funcționale ce descrie în limbaj de scheme bloc funcțiile instalației de automatizare. Pentru

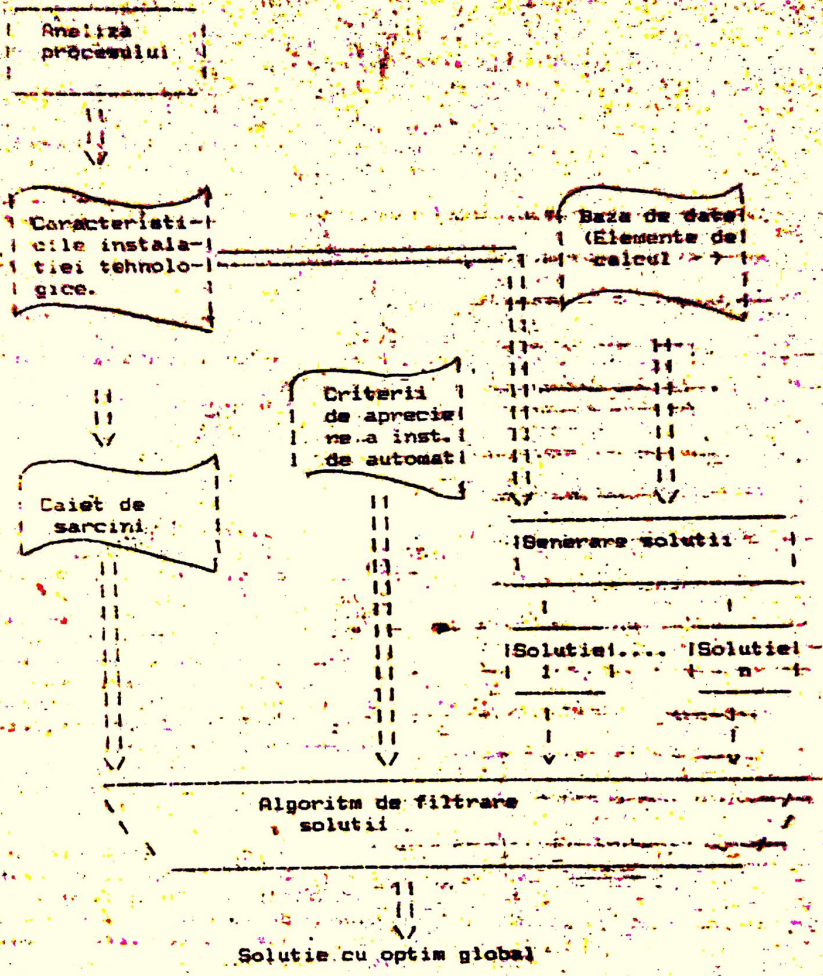


Figura 1. Metodologia pentru configurarea unei instalatii de automatizare cu optim global

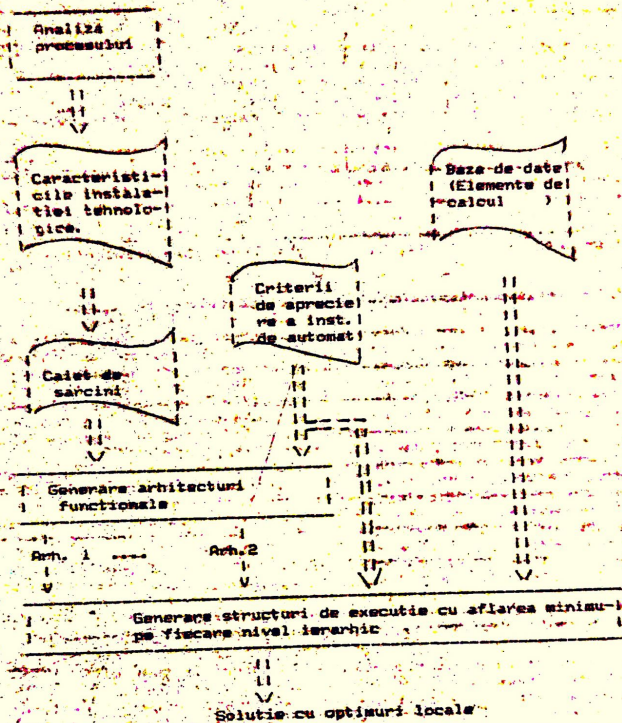


Figura 1. Metodologie pentru configurarea unei instalații de automatizare cu optimizări locale

Instalațiile complexe, arhitectura este distribuită și pe mai multe niveluri. De asemenea, întotdeauna sînt posibile mai multe arhitecturi functionale, în această fază a proiectării neexistînd suficiente date pentru discriminarea celei mai bune soluții. Proiectarea structurii de execuție care materializează în limbaj de elemente de automatizare (concrete) arhitecturile functionale. Structurile de execuție folosesc elemente din baza

de date.

Deoarece pentru fiecare funcție există un număr mare de elemente de automatizare în etapa a doua a proiectării există o "explozie combinatorială". Pentru evitarea acestei situații este indicată proiectarea structurilor de execuție cu selectarea optimului local. Aceasta proiectare presupune respectarea următoarelor etape:

1. se elaborează arhitecturile funcționale;
2. se elaborează structurile de execuție după principiul "top-down" sau "bottom-top". Pentru fiecare nivel și structurilor se aplică algoritmi de decizie care țin seama de criterii specifice nivelului. Se reține doar arhitectura care este optimă pe nivelul respectiv.
3. Se reia pasul 2. până la parcurgerea tuturor nivelurilor.

2. Criterii de alegere a elementelor de calcul

Criteriile de apreciere a elementelor de calcul exprimă gradul de adecvare al acestora la cerințele instalației de automatizare sau la pretențiile beneficiarului. Aceste criterii pot fi împărțite în 4 clase:

- a) criterii funcționale - sunt acele criterii care se referă calitativ și cantitativ la sistemul de automatizare, independent de realizarea tehnologică (timpul de răspuns, numărul intrărilor/ieșirilor, tipul marimilor de intrare, etc);
- b) criterii tehnologice - referitoare la adaptarea echipamentului de automatizare la mediul fizic implicat de instalația tehnologică (temperatura de lucru, perturbatii, etc);
- c) criterii operationale - sunt acele aspecte care privesc partea de exploatare a instalației (extindere în spațiu și timp, mentenabilitate, etc.);
- d) criterii economice - sunt acele criterii ce privesc prețul materialelor, al manoperei, al întreținerii, etc;

Pe baza acestor criterii trebuie luate decizii privind soluția optimă atât în cazul global cât și în cazul local. Există două posibilități de a lua decizii referitoare la soluția optimă:

- arbori de decizie bazati pe structuri "if then else"
- folosirea rezultatelor din teoria deciziilor

Deoarece prima posibilitate nu modelează corect realitatea (aprecierea soluțiilor referitoare la un criteriu ar trebui făcută prin da sau nu) și este rigidă (aparitia unei noi soluții ar modifica întregul arbore de decizie) se preferă a doua posibilitate de decizie.

Conform teoriei deciziei, pentru rezolvarea problemei de decizie se construiește tabela de evaluare din figura 3.

Prin $C_1, C_2, \dots, C_n$ se înțeleg criteriile de apreciere iar prin $A_1, A_2, \dots, A_m$ se înțeleg alternativele (soluțiile) care fac obiectul deciziei. Nu toate criteriile sunt la fel de importante relativ la aplicație pentru care se dorește alegerea soluției optime de aceea fiecareia i se atribuie o pondere w_j care poate fi un număr real cuprins între 0 și 1 sau o multime vagă.

	Ponderi	A1	A2		A _m
C1	w1	r11	r12		r1m
C2	w2	r21	r22		r2m
...	...	...	...	...	...
Cn	wn	rn1	rn2		rn _m

Figura 3.

Gradul în care soluția A_j satisface criteriul C_i se apreciază prin numere reale cuprinse între 0 și 1 sau prin mulțimi vagi (rate).

Prezentăm în continuare un algoritm pentru selectarea soluției optime. Acest algoritm se aplică atunci când ratele r_{ij} ca și ponderile w_i nu mai sînt valori reale ci mulțimi vagi. Situația aceasta se întîlneste în practică atunci când există dubii în legătura cu valorile exacte ale ponderilor și ratelor. În această situație gradul de satisfacere de către varianta A_j a criteriului C_i este reprezentată prin funcția de apartenență $u_{p_{ij}}(r_{ij})$, unde r_{ij} ia valori pe intervalul $[0, 1]$, $u_{p_{ij}}$ luînd valori, de asemenea, tot în același interval.

Ca un exemplu, să considerăm că nu există informații suficiente sau o metodă exactă de determinare a r_{ij} ca o valoare numerică. În acest caz se acordă lui r_{ij} mai multe valori, fiecare valoare fiind însoțită de altă valoare $u_{p_{ij}}(r_{ij})$ care exprimă gradul de încredere în valoarea r_{ij} . Se poate obține, de exemplu, mulțimea $r_{ij} = (0.8/0.7, 0.6/1)$ care exprimă faptul că $u_{p_{ij}}(0.8) = 0.7$ și $u_{p_{ij}}(0.6) = 1$. Deci se are mai mare încredere în valoarea 0.6 alui r_{ij} . Pentru o cit mai corectă reflectare a realității este indicat ca, în aceste situații r_{ij} și $u_{p_{ij}}$ să fie atribuite de mai mulți specialiști.

Aceeași discuție este valabilă și pentru ponderile w_i . Cînd nu se poate aprecia cu siguranță ponderea criteriului C_i se recurge la exprimarea ponderii prin mulțimi vagi.

Algoritmul de determinare a valorii optime este următorul:

$$P_1: u_{z1}(z) = \left[\bigwedge_{j=1}^n u_{w_j}(w_j) \right] \bigwedge \left[\bigwedge_{k=1}^m u_{r_{ik}}(r_{ik}) \right]$$

Simbolul $\bigwedge$ desemnează operația de aflare a minimului iar z desemnează o mulțime $(w_1, w_2, \dots, w_n, r_1, r_2, \dots, r_n)$.

$$P_2: u_{r1}(r) = \sup_{z \in Z} \{ u_{z1}(z) \mid u_{z1}(z) = 1 \} \quad (4)$$

$$P_3: u_{1/r}(1/r_1, \dots, r_m) = \begin{cases} 1 & \text{dacă } r_i = w_j \quad \forall j \in J \\ 0 & \text{altfel} \end{cases}$$

$$P_4: u_{g1}(r_1, \dots, r_m) = u_{r1}(r_1)$$

P5. $u_k(i) = \sup_{r_1, r_2, \dots, r_{n-1}} R(i/r_1, r_2, \dots, r_{n-1})$ pentru $i = 1, 2, \dots, n$.

P6. Varianta optimă va fi cea care are cea mai mare valoare pentru u_1 .

Observație. Ponderile și ratele care nu sînt vagi pot fi transformate în mulțimi vagi cu funcția de apartenență 1.

Bibliografie

1. Baas & Kwakernack "Rating and Ranking multiple aspects alternative", Automatica IFAC, 1985
2. Claude Sourisse "Automatisme: choisir ses technologies de commande", Le Nouvel Automatisme, mai 1984.

SISTEM DE CRIPTARE CU CHEIE PUBLICA PRIN METODA FUNCTIILOR UNISENS DE TIP "RUCSAC"

autor: Ing. C. RADU
Institutul Politehnic-Bucuresti

1. INTRODUCERE

Dezvoltarea sistemelor de "posta electronica" si "transfer bancar electronic" a determinat aparitia sistemelor de criptare cu cheie publica in retelele de calculatoare, pentru a se asigura protej. informatiilor impotriva unor intrusi precum si a partenerilor de retea neavizati sau neautorizati la acces.

Prin sistemul de criptare va trebui sa asiguram doua proprietati importante ale postei clasice:

1. caracterul privat al postei clasice;
2. posibilitatea semnarii mesajelor;

In aceasta lucrare vom prezenta implementarea metodei "rucsacului" de criptare cu cheie publica, metoda ce are proprietatea ca publicarea cheii de criptare a unui mesaj nu face posibila dezvaluirea cheii corespunzatoare de decriptare decit pentru utilizatorul autorizat, posesor al unei informatii suplimentare secrete, numita trapa.

2. FUNCTII UNISENS CU TRAPA. PROPRIETATILE SISTEMELOR DE CRYPTARE CU CHEIE PUBLICA

Ideea criptografierii cu cheie publica este strins legata de ideea functiilor unisens cu trapa. Astfel, dindu-se un argument x este usor de calculat valoarea functiei $f(x)$ in timp ce calculul lui x din $f(x)$ este o problema intructabila (nerezolvabila in timp polinomial). Situatiea este prezentata in Fig. 1.

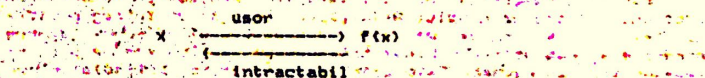


Fig. 1

Ne-am referit la $f(x)$ ca la o functie, dar ea trebuie intelesă in sensul mai larg al unei metode de criptare nondeterministe. Mai mult, calculul lui x din $f(x)$ este o problema intructabila doar pentru un criptanalist neautorizat, dar trebuia sa fie simpla pentru receptorul legal, care este

posesia unei anumite informații private denumită în mod sugestiv trapa.

Conceptul prezentat anterior face posibil ca într-un sistem de criptare cu cheie publică, fiecare utilizator să-și plaseze procedura de criptare E într-un fișier accesibil tuturor, adică acesta să prezintă ca un director ce conține numele utilizatorului și cheia folosită de el. Dar utilizatorul păstrează secretul asupra procedurii de decriptare D corespunzătoare. Procedura E aparține clasei funcțiilor unisens cu trapa și are următoarele proprietăți :

a) Decriptarea formei criptate a unui mesaj M conduce la obținerea mesajului M . Formal :

$$D(E(M))=M ;$$

b) Atît D cît și E sînt proceduri ușor de calculat;

c) Prin relevarea publică a metodei de criptare E utilizatorul nu dezvaluie o cale ușor de urmat în vederea calculului lui D .

d) Dacă un mesaj M este mai întîi decriptat și apoi criptat, M este rezultatul obținut. Formal:

$$E(D(M))=M ;$$

Cînd utilizatorul dezvaluie metoda de criptare, el furnizează o metoda foarte ineficientă de calcul al lui $D(C)$ (unde C este mesajul de criptat): testarea tuturor mesajelor M pînă cînd este găsit acela astfel încît $E(M)=C$. Dacă proprietatea c) este satisfăcută numărul mesajelor ce trebuie astfel testate va fi atît de mare încît timpul de calcul afectat face problema intratabilă.

Proprietatea d) este necesară doar pentru implementarea semnăturilor. Oricum mesaj este textul criptat al unui alt mesaj și orice text criptat este el însuși un mesaj permis.

Metoda rucsacului de criptare se bazează pe următoarea problemă: se dau un n -uplu $A=(a_1, a_2, \dots, a_n)$ și un întreg pozitiv k . Intuitiv k indică mărimea rucsacului și fiecare dintre numerele a_i reprezintă mărimea articolelor particulare ce pot fi împachetate în rucsac. Este clar că cunoașterea articolelor ce intra în rucsac conduce imediat la aflarea mării sale k în timp ce cunoașterea mării rucsacului nu permite afirmarea articolelor ce îl compun. Din acest motiv metoda rucsacului poate fi privită ca o funcție unisens cu trapa, trapa informațională fiind reprezentată de cunoașterea setului de articole ce pot printr-un subset să completeze rucsacul.

3. CRIPTAREA/DESCRIPTAREA PRIN METODA "RUSSACULUI".

Vom începe cu enunțarea unui set de concepte utilizate.

Un vector "russac" este un vector ordonat de întregi pozitivi distincti a_i astfel $a_1, a_2, \dots, a_n$.

O exemplificare a problemei "russacului" este perechea (A, k) unde A este un vector "russac" iar k este un întreg pozitiv (cu sensul de "continut" al russacului).

O soluție pentru (A, k) este un subset al lui A , ale cărui elemente sumate dau k . Deoarece vorbim despre un subset fiecare a_i apare în suma cel mult odată. De aceea problema "russacului" este uneori denumită și problema sumei subsetului.

Cea mai comună variantă a problemei este de a spune despre o pereche (A, k) dacă are sau nu soluție. O variantă folosită în criptografie este de a produce o soluție pentru o pereche (A, k) dată, când se cunoaște existența soluției.

Criptarea constă în folosirea unui vector "russac" A pentru a cripta un bloc C de n biți, prin sumarea componentelor lui A ce apar în poziția corespunzătoare lui 1 din C . Dacă suma este notată prin k , atunci decriptarea ridică problema găsitii lui C din k , sau a găsitii lui C din A și k dacă avem criptosisteme cu cheie publică. Echivalent dacă pe C îi vedem ca un vector coloană de n biți, atunci criptarea constă în calculul produsului $k = A \cdot C$. Ca o exemplificare să luăm $n=6$ și $A = (3, 4, 1, 5, 1, 2)$. Atunci mesajul $C = (1, 1, 0, 0, 1, 0)$ este criptat astfel:

$$k = (3, 4, 1, 5, 1, 2) \cdot (1, 1, 0, 0, 1, 0)^T = 65$$

Pentru acest vector "russac" A , toate criptotextele k sînt numere ≤ 81 , corespunzînd unui $C = (1, 1, 1, 1, 1, 1)$.

Deoarece unicitatea decriptării este de dorit, vectorii "russac" A ar trebui să aibă proprietatea că pentru orice k , toate perechile (A, k) posedă cel mult o soluție. Astfel de vectori "russac" A sînt referiți ca injectivi. Terminologia este foarte naturală, deoarece injectivitatea lui A determină ca și funcția indusă de A să fie injectivă.

Pentru anumiți vectori A , toate perechile (A, k) sînt ușor de rezolvat. Vom vedea că vectorii super-crescători posedă o astfel de proprietate.

Un criptosistem cu cheie publică se poate baza pe astfel de vectori într-o manieră clară: Ambii parteneri, emițatorul și receptorul cunosc vectorul A . Pe de altă parte dacă un vector B este publicat ca cheie de criptare, atunci receptorul legal trebuie să aibă o informație trează secretă, pentru transformarea lui B și a criptotextului într-o problemă ușoară a "russacului". Vom arăta cum se poate efectua aceasta utilizînd același tip de vectori super-crescători. Construcția va fi dată mai detaliat.

Un vector "rucsac" $A = (a_1, a_2, \dots, a_n)$ este crescator respectiv super crescator daca :

$a_j \leq a_{j-1}$ respectiv $a_j \leq a_1$ pentru $j=2, 3, \dots, n$

Este evident ca orice vector super-crescator este si crescator. Pentru un vector "rucsac" A definim:

$$\max A = \max(a_j, j=1, 2, \dots, n)$$

Fie x un numar pozitiv. Vom nota cu $[x]$ partea intreaga a lui x care se bucura de proprietatea ca este cel mai mare intreg mai mic sau egal cu x . Pentru intregii x si m , cu $m \neq 0$ vom nota cu $(x, \text{mod } m)$ cel mai mic rest pozitiv al lui x modulo m . Este usor de verificat ca au loc urmatoarele doua relatii:

$$(x, \text{mod } m) = x - [x/m] * m$$

$$x = (x, \text{mod } m) + [x/m] * m$$

Vom defini acum doua variante ale notiunii de multiplicare modulara. Sa consideram un vector "rucsac" A, un intreg $m \neq 0$ si un intreg pozitiv $t \neq 0$ astfel incit cel mai mare divizor comun al numerelor t si m este 1 (adica t si m sint prime).

Daca $B = (b_1, b_2, \dots, b_n)$ este un vector astfel incit:

$$b_i = (t * a_i, \text{mod } m)$$

vom spune ca B rezulta din A prin multiplicare modulara cu respectarea modulului m si a multiplicatorului t , sau mai prescurtat cu respectarea perechii (m, t) .

Conditia $\text{c.m.m.d.c.}(t, m) = 1$ garanteaza existenta elementului invers $u = t^{-1}$ astfel incit :

$$t * t^{-1} = t * u = 1 \pmod{m} \text{ cu } 1 \leq u < m$$

Aceasta implica ca si A rezulta din B tot prin multiplicare modulara cu respectarea lui m si u (unde $m \neq 0$ max B, deoarece elementele b_i sint reduse modulo m).

Daca in locul conditiei $\text{c.m.m.d.c.}(t, m) = 1$ vom inlocui-o cu conditia mai puternica:

$$m \nmid a_i$$

vom spune ca B rezulta din A prin multiplicare modulara puternica cu respectarea lui m si t .

Sa observam ca acum nu putem concludiza ca A rezulta din B prin multiplicare modulara puternica cu respectarea lui m si u deoarece inegalitatea:

$$m \mid b_1$$

nu mai are loc in mod necesar. Desigur insa ca A rezulta din B prin multiplicare modulara cu respectarea lui m si u.

Un realizator de criptosistem alege componentele de proiectare A, t, m, B astfel:

- 1) A vector rucsac super-crescator;
- 2) B vector realizat din A prin multiplicare modulara puternica cu respectarea lui m si t, unde m si t au fost alese ca pentru multiplicarea modulara puternica.

Vectorul B este publicat ca cheie de criptare si blocuri C de cite n biti sint trimise spre utilizator ca numere k, obtinute din B, in aceiasi modalitate cu k:

$$k' = B * C^T$$

Un ascultator neneautorizat trebuie sa rezolve problema (B, k') a "rucsacului". Receptorul autorizat calculeaza $k = u * k \pmod{m}$ si rezolva problema (A, k) . Mecanismul de functionare este dat de lema:

LEMA: Sa presupunem ca $A = (a_1, a_2, \dots, a_n)$ este un vector super-crescator si B rezulta din A prin multiplicare modulara puternica cu respectarea lui m si t. Sa presupunem mai departe ca $u = t \pmod{m}$, k este arbitrar si $k = (u * k, \pmod{m})$. Atunci au loc urmatoarele rezultate:

1. Problema "rucsacului" (A, k) este rezolvabil in timp liniar. Daca o solutie exista este unica.
2. Problema rucsacului (B, k') are cel mult o solutie.
3. Daca o solutie a lui (B, k') exista, ea o egaleaza pe unica solutie a problemei (A, k) .

DEMONSTRATIE

1. Orice problema a rucsacului cu un vector A super-crescator poate fi rezolvata in timp liniar prin citirea in A o singura data de la dreapta la stanga, rezultind o solutie unica.

2. Sa presupunem ca un vector de n biti D este o solutie a problemei (B, k') , adica:

$$k' = B * D^T$$

Atunci avem:

114

$$\begin{aligned}
 k &= u \cdot k' \pmod{m} = u \cdot B \cdot D^T \pmod{m} = \\
 &= u \cdot (t \cdot A \cdot D^T) \pmod{m} = t^{-1} \cdot t \cdot u \cdot (A \cdot D^T) \pmod{m} = \\
 &= A \cdot D^T \pmod{m}.
 \end{aligned}$$

Deoarece m excede suma componentelor din A , trebuie să avem $A \cdot D^T \pmod{m}$. Deoarece și $k \pmod{m}$, prin definiția lui k , concluzionăm că $k = A \cdot D^T$. Astfel D egalează și soluția unică a lui (A, k) . Aceasta demonstrează 3. Deoarece am plecat cu o soluție arbitrară pentru (B, k) și am arătat că egalează unică soluție alui (A, k) am demonstrat de fapt și pe 2.

În continuare vom ilustra această metodă de criptare-decriptare pe un vector cu 10 componente pentru a asigura citirea ușoară a exemplului.

4. EXEMPLU

Fie un vector "rucsac" A cu $n=10$ componente de tip super-crescător dat după cum urmează:

$$A = (103, 107, 211, 430, 863, 1718, 3449, 6907, 13807, 27610)$$

Alegem $m = 55207$, a_i și $t_1 = 25236$ cu proprietatea că c.m.m.d.c. $(t, m) = 1$. Atunci $t^{-1} = u = 1061$. Prin multiplicare modulară puternică vom obține componentele lui B astfel:

$$103 \cdot 25236 \pmod{55207} = 103 \cdot 25236 - \lfloor 103 \cdot 25236 / 55207 \rfloor \cdot 55207 = 4579$$

$$107 \cdot 25236 \pmod{55207} = 107 \cdot 25236 - \lfloor 107 \cdot 25236 / 55207 \rfloor \cdot 55207 = 50316$$

vectorul B devenind:

$$B = (4579, 50316, 24924, 30908, 27110, 17953, 32732, 16553, 22075, 53620) = (t \cdot a_i \pmod{m})$$

Vectorul B este cheia publică pentru criptare în timp ce A, t, u, m constituie trapa secretă. Cunoașterea lui m sau t sau u facilitează aflarea imediată a celorlalte articole, însemnând practic dezvăluirea secretului.

Să folosim acum cheia publică B pentru cifrarea textului clar: **BUN VENIT LA IASI**. Pentru aceasta vom adopta o codare numerică în care blank-ul are codul 00000, iar literalele A-Z valori între 1-26. Aceste valori vor fi codificate pe 5 biți (32 de cuvinte sint acoperitoare). Astfel:

blank 00000


```

      A      00001
      B      00010
      .
      .
      .
      Z      11010
  
```

Deoarece B poate fi folosit pentru criptarea blocurilor de 10 biti, textul clar trebuie sa fie impartit in blocuri constind din 2 caractere fiecare. In acest mod criptotextul devine:

text clar	codul binar	criptotextul
BU	0001010101	119034
N	0111000000	106148
VE	1011000101	130584
NI	0111001001	192500
T	1010000000	29503
LA	0110000001	128860
I	0000001001	86352
AS	0000110011	120758
I	0100100000	77426

Criptotextul pentru BU a fost obtinut din inmultirea:
 $A \cdot (0001010101) = 30908 + 17953 + 16553 + 53620 = 119034$
 iar pentru celelalte grupuri de litere s-a procedat similar.

Decriptam acum primul numar 119034. Sa notam ca se efectueaza, mai intai o multiplicare modulara cu $u=1$:

$$k = u \cdot k \pmod{m} = 1061 \cdot 119034 \pmod{55207} = 36665.$$

Vom considera acum problema rucsacului ($A, 36665$). Solutia este obtinuta prin scanarea lui A o singura data de la dreapta la stanga. Oricind numarul avut la dispozitie este cel putin egal cu componenta scanata din A, vom pozitiona bitul pe 1 si noul numar este obtinut prin scaderea componentei vectorului A din numarul avut la dispozitie. Altfel pozitionam bitul pe 0 si numarul avut la dispozitie ramine nealterat. Rezultatul poate fi exprimat dupa cum urmeaza:

numar	componenta A	bit
36665	27610	1--
9055	13807	0 1
9055	5907	1 1→U
2148	3449	0 1
2148	1718	1--
430	863	0--
430	430	1 1
0	211	0 1→B
0	107	0 1
0	103	0--

Vectorul de biti original, din care rezulta textul clar BU, poate fi citit de pe ultima coloana de la baza spre varf. Deci, decriptarea similara a celorlalte numere vom obtine si restul textului clar.

5. BIBLIOGRAFIE

1. Merkle, R., Secure Communication over Insecure Channels, Commun. of ACM, apr., 1978.
2. Merkle, R; Hellman, M, Hiding Information and Signatures in Trapdoor Knapsacks, I.E.E.E. Trans. on Information Theory, sept. 1978.
3. Risto Salomaa, Public-Key Cryptography, 1990.
4. D.E. Denning. Cryptography and Data Security. Addison-Wesley, 1982.
5. J. Seberry, J. Pieprzyk. Cryptography: An Introduction to Computer Security. Prentice Hall, New York, 1989.

OPERATII GEOMETRICE ASUPRA SOLIDELOR MODELATE PRIN OCTREES

Gorgan Dorian, Miron Aurelian, Rusu Dan
Institutul Politehnic Cluj-Napoca

1. Introducere

Diverse domenii de aplicatie necesita reprezentarea geometrica a obiectelor, fiind una dintre cele mai sugestive metode de dialog intre calculator si operator. Operatiile geometrice asupra corpurilor: generare, translatie, diferenta, intersectie, rotatie, scalare, etc, sint rezultatul operatiilor asupra modelelor acestora in calculator.

Printre modelele geometrice mai des utilizate sint:

- modelul B-rep in care muchiile sint elementele de baza. Acest tip de model implica descompunerea solidului intr-un set de fete de ecuatii cunoscute si o posibilitate de conectare a acestor fete, in general printr-un graf;
- modelul C.S.G. (Constructive Solid Geometry). Solidale sint reprezentate prin descompunerea in solide considerate ca primitive grafice (piramida, cub, sfera, paralelipiped, cilindru), care se combina folosind operatori booleeni pentru construirea obiectului dorit;

- Sistemele de modelare oct-trees sint cele mai potrivite modele pentru metoda divide_and_conquer in spatiu, fata de quad_trees in plan.

In aceasta lucrare s-a ales ultimul model precizat, datorita posibilitatilor de prelucrare a corpurilor (prin operatii booleene), si de efectuare a operatiilor asupra codului ce descrie obiectul.

2. Sisteme de modelare oct-trees

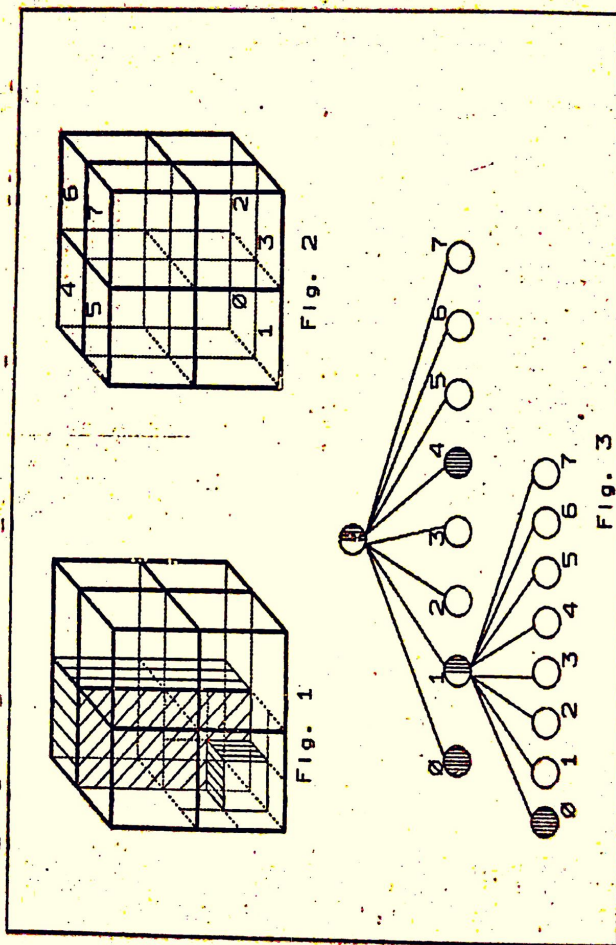
Sistemele cu acest tip de modelare sint larg utilizate in proiectarea asistata, recunoasterea formelor, robotica, etc.

Pentru a crea modelul scenei tridimensionale, se porcesc de la un cub acoperitor pe care il spargem in 8 octanți (Fig. 1).

Un obiect se aproximeaza prin divizare in cuburi mai mici, pina la o dimensiune minima data. Astfel se pot modela obiecte de complexitati diferite. Numerotarea octanzilor este arbitrara, dar aceeași in tot modelul (Fig. 2).

Pentru exemplul din figura 2 se va obtine arborele din figura 3 in care s-au folosit notatiile:

-  plin: tot octantul este umplut cu corpul considerat. In arbore octantul este reprezentat printr-o frunza plina;
-  partial: octantul contine parti din corpul considerat;
-  gol: nu contine parti din obiectul respectiv. In arbore este reprezentat printr-o frunza goala.



Orice nod p va fi în continuare divizat, octuplul rezultat fiind formal reprezentat prin arbore. Procesul poate fi continuat pînă se ajunge la suprafața ce mărginește obiectul. Divizarea se oprește la un nod de dimensiune formal indivizibilă. Pentru fiecare nod din arbore se păstrează următoarele informații:

- x_{min} , y_{min} , z_{min} , x_{max} , y_{max} , z_{max} : coordonatele minime și maxime ale cubului nod;
- frunza : un indicator de tip al nodului, cu valoarea *true* pentru nod frunza și cu valoarea *false* în alt caz;
- *plin* : un indicator de nod plin, avînd valoarea *true* pentru nod plin și *false* în alt caz.

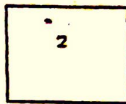
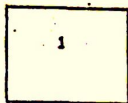
3. Operații asupra arborilor Oct

3.1. Diferența a două corpuri

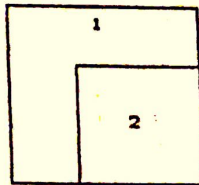
Simplificăm problema și o reprezentăm în plan, cazurile în spațiu fiind aceleași. Analizăm poziția a două cuburi frunza pline aparținînd fiecărui unuia dintre arborii implicați în operația de diferență. Analiza o facem pentru toate combinațiile posibile - pentru fiecare frunza plină din primul arbore, parcurgem al doilea arbore și analizăm poziția sa relativă.

Pot exista următoarele cazuri:

- notăm cu 1 cubul descăzut și cu 2 cubul scăzător.
- a). cuburi disjuncte. Nu se execută nici o operație.



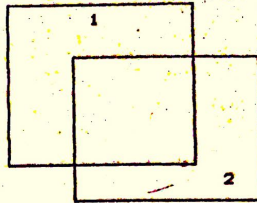
- b). cubul descăzut în interiorul cubului scăzător. Cubul descăzut este eliminat din arborele rezultat.



- c). cubul scăzător în interiorul cubului descăzut. Se va sparge cubul descăzut și se va realiza prelucrarea în cadrul arborelui descăzut.



d). cuburile se suprapun partial.
Se sparg ambele cuburi si se reia prelucrarea din nivelul curent al arborelui descazut.



La cazurile c) si d) prelucrarea se opreste la atingerea unei laturi minime:

- daca muchia cubului descazut este mai mica decit latura minima (sau egala) nu se mai sparge si nu se reia prelucrarea;
- daca muchia cubului scazator este mai mica sau egala cu latura minima se sparge numai cubul descazut si se reia prelucrarea (la cazul d).

3.2. Intersectia a doua corpuri

Intersectia presupune aceleasi operatii ca si diferenta, dar actiunile de la cazurile a) si b) difera:

- in cazul cuburilor disjuncte, primul cub (care anterior era descazutul) devine frunza goala in cadrul arborelui rezultat.
- in cazul in care primul cub este in interiorul cubului al doilea, sau egale, primul cub devine frunza plina in cadrul arborelui rezultat si se trece la urmatoarea frunza plina din primul arbore.

3.3. Translatia unui corp

Se parcurge tot arborele si pentru fiecare nod al sau, terminal sau nu, se aplica operatorul matriceal de translatie, in coordonate omogene:

$$T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ tx & ty & tz & 1 \end{bmatrix} \quad \begin{aligned} P' &= P \cdot T \\ [x' \ y' \ z' \ 1] &= [x \ y \ z \ 1] \cdot T \end{aligned}$$

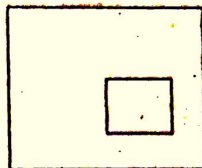
3.4. Rotatia in jurul axelor a unui corp

Se parcurge arborele si pentru fiecare cub ce exista in fiecare nod se reactualizeaza dimensiunile cu operatorii matriceali de rotatie, in coordonate omogene: $R_x(u)$, $R_y(u)$, $R_z(u)$.

3.5. Reuniunea a doua corpuri

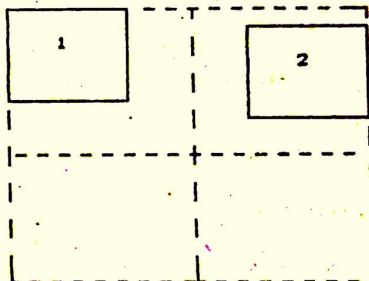
Reuniunea se reduce, in cele din urma, la o intersectie.

a). In acest caz cubul acoperitor este tocmai unul din cuburile initiale (cel mai mare) si tot el este si arborele rezultatant.

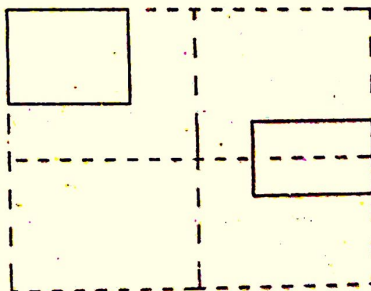


b). Se porneste de la un singur cub acoperitor. Pentru fiecare

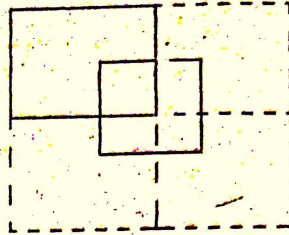
frunza a sa, testam ce cub acoperitor (din cele doua initiale) contine cuburi din cele doua corpuri. Daca nu contine frunza e goala. Daca contine, frunza este egala cu intersectia ei (considerata plina) si



c). operatii la fel ca la punctul b).



d). se porneste de la un cub acoperitor a carui frunza sa fie egala cu cel mai mare dintre cuburile initiale. Aceasta frunza va fi plina. Toate celelalte frunze vor fi intersectiile cu cel de al doilea cub initial.



e). aceleasi operatii ca la punctul d).



4. Concluzii

Programul a fost implementat in limbajul Turbo C, pe un sistem compatibil IBM - AT 286, cu adaptor grafic de tip EGA. Programul este utilizat prin tehnica meniurilor ierarhice, insotite de indicatii (help-uri) de utilizare. Realizeaza functii ca:

- generarea arborilor octree pentru corpuri convexe;
- generarea si verificarea consistentei scenei de obiecte;
- salvarea si incarcarea in memorie a unor arbori deja creati;
- vizualizarea corpurilor modelate prin arbori;
- operarea asupra arborilor;
- dialogul prin meniuri intre utilizator si program.

Arborii rezultati sint destul de mari pentru corpuri complicate si rezolutie mare, dar nu depasesc memoria disponibila. Imbunatatirile urmarite prin modificarea algoritmilor se refera la micorarea timpului de executie. S-au obtinut urmatoarele masuratori ale timpului de executie:

1. Dacă latura cubului minimă este fixă și egală cu 20 :

Cub acoperitor	Țimp construire	Țimp vizualizare
40	-	-
80	3"	4"
120	9"	28"
160	14"	30"
200	44"	10' 7"

Țimpii de construire nu deranjează, corpul odată construit, putând fi încărcat de pe disc. Țimpii care deranjează sunt cei de vizualizare.

2. Testul s-a făcut pentru o piramidă triunghiulară dreaptă cu vârful în origine. Cubul acoperitor este fix cu latura 80.

Latura minimă	Țimp construire	Țimp vizualizare
5	2' 3"	9' 50"
10	12"	28"
20	3"	4"
40	-	-

Din punct de vedere al utilizatorului arborii sunt identificați prin nume. Implementarea s-a făcut astfel încât la un moment dat pot exista maxim 8 arbori în memorie. Pentru a se opera asupra obiectelor acestea trebuie să aibă arborii corespunzători încărcat în memorie.

5. Bibliografie

1. Shi-Kuo Chang, Principles of Pictorial Information System Design, Englewood Cliffs, New Jersey: Prentice-Hall, inc., 1989.
2. Foley, J. Van Dam, A. Fundamentals of Interactive Computer Graphics, Addison Wesley, 1983.
3. Dogaru, D. Elemente de grafica 3-D, Editura Stiintifica si Enciclopedica, Bucuresti 1988.

REPREZENTAREA GEOMETRICA A MODELELOR TOPOLOGICE ARBORESCENTE

Autori: Gorgan Dorian, Albert Laszlo, Rusu Dan
Institutul Politehnic Cluj-Napoca

REZUMATI:

Obiectul acestei comunicări este prezentarea metodei de definire a modelelor topologice sub formă arborescentă și a unei aplicații a acesteia în cartografierea peșterilor.

Un astfel de model poate fi aplicat și în alte domenii de activitate, cum ar fi:

- căi ferate
- drumuri
- rețele de aprovizionare
- cartografie

Aplicația prezentată este astfel implementată încât să rezolve și problemele ce derivă din metodele de efectuare a măsurătorilor în cartografierea peșterilor.

Reprezentarea grafică a hartilor este făcută tridimensional, avantajele acestei metode fiind prezentate în lucrare.

Datele necesare construirii reprezentărilor grafice sînt gestionate cu ajutorul unor programe dBASE.

PREZENTAREA MODELULUI

Prelucrarea informațiilor topologice pe calculator impune reprezentarea lumii reale prin modele relativ complexe. Modelul cel mai general pentru reprezentarea legăturilor dintre obiecte este cel de tip rețea. În general, modelul de tip rețea se poate transpune într-un model ierarhic arborescent fără pierderea de informații esențiale referitoare la topologie.

Informațiile sînt asociate atât nodurilor cît și legăturilor din model.

Legătura poate conține următoarele informații: capacitatea de transport, direcția de circulație, număr de benzi, debit, tipul de legătură (drum, pod, tunel, etc.), forma legăturii (de exemplu dreaptă sau arc).

Nodurile trebuie să conțină obligatoriu informații despre formă și poziție pentru a putea permite reprezentarea geometrică a modelului topologic. În plus, nodurile mai pot conține informații referitoare la aspect (culoare, simbol), nume, date economice, obiective existente (muzee, restaurante, puncte turistice), etc.

Trebuie remarcat faptul că majoritatea nodurilor nu vor conține informații semnificative, ele fiind introduse din alte rațiuni. De exemplu dacă între două noduri legătura nu este omogenă (trecerea printr-un tunel), este necesară introducerea unor noduri noi, astfel încât toate legăturile să fie omogene. Deoarece informațiile geometrice sînt continute în noduri, schimbarea direcției legăturii sau detalierea topologiei vor duce de asemenea, la apariția unor noi noduri.

Atributele legăturilor pot fi înglobate în noduri fără a reduce generalitatea modelului. Se asigură astfel o omogenitate mai mare în tratarea problemei. Legăturile vor conține deci doar informații structurale ce reflectă modul de interconectare a nodurilor. Avînd în vedere cele prezentate mai sus, deosebim două tipuri de noduri:

- Noduri informaționale (I) care conțin informații necesare aplicării modelului într-un anumit domeniu, precum și informații geometrice și cele care reflectă natura și proprietățile legăturii la nodul anterior.

- Noduri geometrice (G) care conțin doar informații geometrice și proprietăți ale legăturii la nodul anterior.

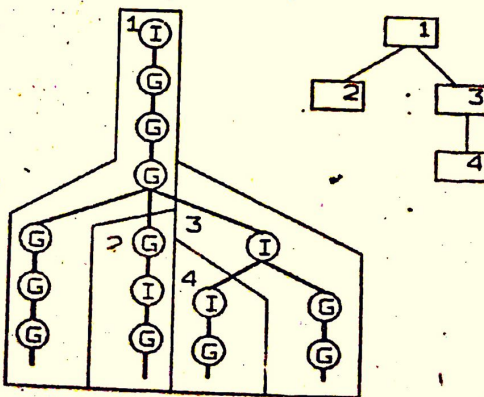


Fig. 1

Urmărind modelul din figură 1 se poate observa existența unor tronșoane (ramuri) interconectate. În cadrul unui tronșon există atât noduri informaționale cât și geometrice. Oricare dintre aceste noduri poate reprezenta un punct de racordare pentru un nou tronșon.

În acest mod de reprezentare a modelului, fiecare tronșon devine un șir de noduri. Deși punctele de racordare aparțin tronșonului superior, ele pot caracteriza în același timp tronșonul inferior.

Pentru reprezentarea grafică a topologiei se poate folosi un model bidimensional sau unul tridimensional.

Deși modelul bidimensional are avantajul simplității, noi am optat pentru cel tridimensional, acesta prezentând o serie de avantaje cum ar fi:

- reprezentarea cât mai fidelă a lumii reale
- posibilitatea reprezentării din diferite puncte de observație;
- generarea unor proiectii pentru construirea de hărți;
- obținerea unor informații de tip "diferență de nivel", inaccesibile în cazul hărților obișnuite.

APLICATIE

Reprezentarea pesterilor este unul dintre domeniile în care modelarea arborescentă a topologiei poate fi aplicată cu succes. Aplicarea acestui model derivă în mod natural din modul în care speologii efectuează cartografierea și măsurătorile într-o pesteră. Măsurătorile pornesc de la intrare și sînt efectuate de galerii. Galeriele pot fi asociate tronșoanelor prezentate în modelul teoretic.

Informațiile geometrice prezente în noduri vor lua o formă specială ce derivă din modul de efectuare a măsurătorilor. Astfel, pentru fiecare nod măsurătorile se fac în linie dreaptă, relativ la nodul anterior. Se înregistrează următoarele informații referitoare la dreapta care unește cele două noduri:

- a) mărimea (distanța dintre cele două noduri)
- b) azimutul (deviația față de direcția NORD)
- c) înclinatia față de orizontală.

Acest mod se poate asimila cu un model avînd coordonate sferice, relative.

O problemă aparte o reprezintă modelarea încăperilor (săliilor) de diverse forme. Culegerea informațiilor cu privire la lățimea și înălțimea galeriilor este foarte dificilă și ar duce la apariția a foarte multe noduri datorate legăturilor neomogene.

Rezolvarea acestei probleme se face prin efectuarea unor măsurători suplimentare numite "radiații". Dintr-un nod al tronsonului se pot realiza mai multe măsurători radial, până la puncte extreme ale sălii, astfel încât circumferința acestor radiații să reflecte dimensiunile și forma sălii.

În această reprezentare a modelului fiecare sală devine un sir de noduri geometrice. Legătura va avea atașată caracteristica "radiație" pentru a diferenția nodurile radiate de cele de tip origine, acestea din urmă participând efectiv la calculul coordonatelor carteziene absolute ale nodurilor.

Datele referitoare la noduri sînt memorate și manipulate folosind programe dBASE și sînt exploatate din Turbo Pascal. Astfel se pot folosi pentru crearea și întreținerea fișierelor de date posibilitățile deosebite ale dBASE, iar prelucrarea datelor este făcută în Turbo-Pascal, folosind facilitățile grafice oferite de acest mediu de programare.

Interfața cu utilizatorul este realizată pe bază de meniuri. Din meniul principal se poate selecta modul de lucru al programului: afișare arbore de structură (folosit mai ales în verificarea datelor introduse) sau afișarea grafică a hărții rezultate în urma prelucrării datelor.

Există de asemenea posibilitatea de a încărca, modifica și salva un fișier de date, toate putînd fi executate în mod de afișare a structurii.

Pentru reprezentarea în mod grafic a hărților există posibilitatea de a alege planul de proiecție sau alegerea unui punct de observație pentru o proiecție ortogonală. Înainte de a putea face reprezentarea grafică a structurii este necesară cunoașterea sau calcularea coordonatelor carteziene ale punctelor de derivatie. Dacă în cursul calculării acestor coordonate apar erori acestea sînt semnalate printr-un mesaj ce conține informații despre tipul erorii și locul (din structură) unde a fost detectată.

CONCLUZII

Deși aici este prezentat un exemplu de folosire a modelării topologice arborescente, cu un cîmp de utilizare relativ restrîns, principiile care stau la baza acestei lucrări pot fi folosite cu rezultate deosebite și în realizarea unor hărți sau diagrame legate de rețele de transport, drumuri, rețele hidrografice, etc.

Mapa de pescaria



REALISMUL VIZUAL AL REPREZENTĂRILOR GRAFICE GRAFTALI

autor: asist. ing. IVAN FARKAS
Univ. Tehnica Timisoara

1. INTRODUCERE

Majoritatea scenelor grafice 3D care urmează a fi reprezentate pe calculator pot fi descrise utilizând primitivile grafice asociate unor elemente geometrice ca segmente de dreapta, arce de cerc, poligoane sau curbe de aproximare și de interpolare. Rezultatele obținute sînt de cele mai multe ori multumitoare, atunci cînd o aproximare a scenei grafice reale este suficientă. Dacă se dorește însă reprezentarea unei scene grafice ce simulează aspecte ale mediului natural exterior, de exemplu mînti, arbori, nori etc., cu un grad ridicat de realism vizual al reprezentării, utilizarea primitivelor grafice menționate este aproape imposibilă datorită necesității de a defini un număr impresionant de puncte ale proiecției scenei grafice, precum și relațiile geometrice dintre acestea. Soluționarea acestei probleme în acest cadru devine deosebit de laborioasă, în mod special atunci cînd se dorește reproducerea nu numai a formelor dar și a culorilor și a texturilor.

Reprezentarea unei scene grafice utilizînd curbe și poligoane va genera imagini cu un aspect nenatural pentru că acestea sînt prea precise, prea regulate și prea izbutite. Obiectele scenei grafice vor avea un aspect mult mai apropiat de cel natural dacă ar putea fi reprezentate cu un grad oarecare dar controlat de iregularitate.

Incorporarea iregularităților în reprezentările grafice implică utilizarea unor tehnici de modelare și transpunere grafică specializate. Între acestea de o largă răspîndire se bucură datorită efectelor realiste, tehnicile bazate pe curbile și suprafețele fractale. Rezultatele generate în aceste condiții sînt deseori spectaculoase.

Un fractal poate fi considerat un obiect a cărui formă este neregulată la orice scară, indiferent de gradul de detaliere la care este observat, spre deosebire de formele standard ale geometriei euclidiene. Fractalii au o dimensiune fracțională.

Tehnicile de programare utilizînd fractalii oferă o cale de generare a unor cantități enorme de date grafice pornind de la un set relativ redus de reguli de înlocuire. Aceste tehnici pot fi considerate ca un set de operatori care formează un așa zis amplificator de bază de date grafică. În același cadru au fost dezvoltate și alte tehnici specifice care își găsesc aplicații în reprezentarea multor obiecte din natură. Între acestea tehniciile bazate pe graftali și sisteme de particule sînt deosebit de interesante. [2]

2. GRAFTALI

Noțiunea de graftal a fost introdusă de Alvy Ray Smith. Atunci cînd în scenele grafice apar obiecte grafice ce modelează plante, arbori, arbusti etc., este propice utilizarea graftalilor.

Un obiect modelat ca graftal este construit ca un sir de simboluri de exemplu 0, 1, I, J. Semnificația acestor simboluri este dependentă de tipul obiectului modelat. În cazul unui arbore

semnificatie acordata simbolurilor poate fi urmatoarea:
 1 reprezinta un segment de culoare maro a unei ramuri;
 0 reprezinta un segment de culoare verde a unei ramuri
 I reprezinta un punct de ramificatie in arbore si precizeaza inceputul unei noi ramuri
 J reprezinta sfirsitul unei ramuri in arbore
 Simbolul I este apoi interpretat alternativ ca ramificatie la stanga respectiv la dreapta.

Semnificatie acordata simbolurilor 1 respectiv 0 poate fi modificata, de exemplu pentru a indica ramuri de grosimi diferite sau iramurarea Ofruct etc.

Reprezentarea obiectului se poate face simplu cu ajutorul unei proceduri care accepta la intrare un sir de simboluri corect din punct de vedere sintactic, executa comenzile implicate de semnificatia fiecarui simbol si produce imaginea obiectului pe dispozitivul grafic de iesire.

Unghiul cu care se incepe desenarea unei noi ramuri la intalnirea in sirul de intrare a simbolului I poate fi fixat sau poate varia aleator intr-un anumit domeniu de valori. In ultimul caz imaginile obtinute sint mult mai sugestive si diverse.

Actiunile executate la intalnirea simbolurilor 0 respectiv 1 sint in esenta doua:

- se stabileste culoarea de trasare(sau caracteristica codificata)

- se traseaza un segment de dreapta cu o anumita lungime in directia precizata de unghiul curent de trasare. Acest unghi se stabileste in momentul interpretarii simbolului I. Lungimea segmentului poate fi fixata sau poate fi variata aleator intr-un domeniu de valori.

Interpretarea simbolului I conduce la modificarea unghiului de trasare cu valoarea specificata.

Intalnirea in sirul de intrare a simbolului J precizeaza sfirsitul unei ramuri in arbore si implica readucerea pozitiei curente de trasare in punctul in care se afla inaintea ultimului simbol I intalnit. Totodata se readuce unghiul de trasare la valoarea din acel moment.

Actiunile induse de interpretarea simbolurilor I respectiv J necesita retinerea unor parametri de control ai executiei la fiecare nivel de ramificare al arborelui. Fiecare ramura din arbore este la rindul ei un arbore, asemanator intregului, aparind in pozitii diverse si cu lungimi diferite. Aceasta observatie conduce in mod natural spre utilizarea unei metode de implementare bazata pe tehnica programarii recursive si a primitivelor grafice de iesire in coordonate relative.

C. GENERAREA GRAFTALILOR

Generarea sirului de simboluri se face din aproape in aproape utilizind un set redus de reguli de inlocuire aplicate in mod repetat asupra unui sir initial. Sirul de intrare este parcurs secvential incepind cu primul simbol ignorind parantezele. Fiecare simbol 0 respectiv 1 va genera un nou subsir de simboluri conform cu regulile de inlocuire precizate, iar acesta se va insera in sirul de iesire. Simbolurile I si J se insereaza in sirul rezultat in pozitia curenta, fara nici o modificare.

Un exemplu de reguli de înlocuire este:

0 → 110111010

1 → 11

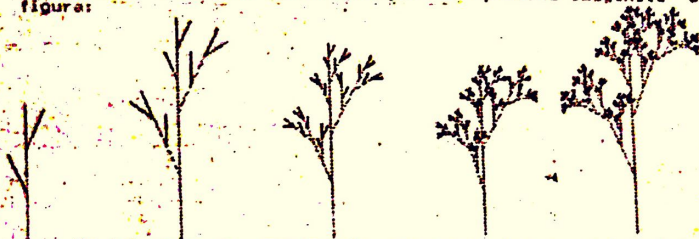
Considerind ca sirul initial de pornire este S1: 0, aplicind în mod repetat regulile de înlocuire vom obtine:

S11: 0 → 110111010

S12: 110111010 → 1111101110101111101110101110111010

Un numar de aproximativ 7 repetari ale procesului de înlocuire este suficient pentru obtinerea unor imagini sugestive.

Fixind modificarea valorii unghiului de trasare la $\pi/6$ traversarea sirurilor S11, S12, ..., S15 vor produce imaginile din figura:



Se observa ca segmentele corespunzatoare simbolului 0 reprezinta ramuri terminale in arbore. In acest exemplu este posibil ca semnificatia acordata simbolurilor 0 si 1 sa fie modificata astfel

1 reprezinta ramura

0 precizeaza pozitia de amplasare a unui obiect grafic reprezentind o floare sau o frunza

Imagina obtinuta in acest caz este desigur mai sugestiva, cu grad sporit de realism vizual.

Calitatea reprezentarii grafice poate fi imbunatatita prin introducerea unui grad de nedeterminism in model. In acest caz fiecare din simbolurile 0 respectiv 1 poate primi o semnificatie multipla. Substituirea simbolurilor cu una din semnificatiile concrete se va face in mod aleatoriu cu o anumita probabilitate controlata. Avind proceduri de interpretare adecvate se pot obtine imagini deosebit de sugestive.

Marimea obiectului generat depinde de numarul de repetari ale regulilor de înlocuire utilizat in procesul de generare precum si, desigur, de lungimea fizica a segmentelor sau a obiectelor produse ca urmare a interpretarii simbolurilor 0 respectiv 1.

Regulile de înlocuire (1) sint aplicate oricand este intilnit in sirul de intrare unul din simbolurile 0 si/sau 1. Aceasta forma a regulilor de înlocuire se numeste independenta de context.

Un alt exemplu interesant de reguli de înlocuire independente de context este:

0 → 1

1 → 11011011

Pornind de la sirul initial S1:0 ele genereaza nervurile unei frunze, la primul nivel se substitutie.

O alta clasa de graftali se obtine pornind de la un set de reguli de inlocuire dependente de context. In acest caz, subsirul de simboluri, care inlocuieste un simbol curent din sirul de intrare, depinde de simbolurile precedent si urmator ale simbolului curent. Avind doua simboluri substituibile rezulta un set de opt reguli de inlocuire dependente de context. De exemplu:

```
000 → 0
001 → 1
010 → 0
011 → 1
100 → 0
101 → 001011
110 → 0
111 → 0
```

Aplicarea acestor reguli de inlocuire necesita precizarea actiunilor in cazurile limita cind codurile de trei simboluri adiacente nu pot fi formate, si anume:

- daca nu exista simbol precedent (la inceputul sirului sau a unei ramuri) se va adauga formal un simbol 1 la stinga pozitiei curente
- daca nu exista simbol urmator (la sfirsitul sirului sau a unei ramuri) se va adauga formal un simbol 1 la dreapta pozitiei curente

Pornind de la sirul SI:0 se vor obtine sirurile:

```
SI1: 0 → 001011
SI2: 001011 → 0110010111
SI3: 0110010111 → 0010111011001011101
```

Traversarea sirurilor obtinute vor produce imaginea unui arbust aplecat in bataia vintului. Inlocuind regula 100 cu 1100 → 10111 imaginea obtinuta se modifica semnificativ.

4. FRACTALI SI GRAFTALI

Fractalii si graftalii sînt obiecte grafice inrudite. Cu toate ca graftalii nu sînt fractali deoarece nu contin un grad infinit de detalieri ei sînt adeseori numiti subfractali. Setul de reguli de inlocuire care stau la baza generarii graftaliilor poate fi extins pentru generarea unor fractali artificiali. Secvente de comenzi grafice in coordonate relative pot fi reprezentate prin siruri compacte construite din simboluri si valori numerice despartite printr-un separator:

T<1> → traseaza un segment de lungime <1> in directia curenta de trasare

M<u> → modifica incremental directia curenta de trasare cu valoarea <u>

In acest context regula de inlocuire

T<1> → T<1/3>;M<60>;T<1/3>;M<-120>;T<1/3>;M<60>;T<1/3> va genera curba Koch triadica.

Bibliografie

1. Pokorny, Gerald-Computer Graphics: The Principles Behind The Art and Science
2. Rankin-Computer Graphics Software Construction

REDAREA UMBRELOR IN SCENE GRAFICE 3D MODELATE PRIN REȚELE DE POLIGONE

autorii: asist.ing. IVAN FARKAS
ing. GEORGETA FARKAS
Universitatea Tehnică Timișoara

1. Introducere

Realismul vizual al reprezentărilor scenelor grafice în grafica pe calculator este un deziderat, și nu de puține ori o necesitate. În contextul actual al penetrării masive și susținute a graficii pe calculator în domenii de o largă diversitate, stacheta exigentei relative la calitatea reprezentărilor este din ce în ce mai ridicată. Reprezentările grafice, pe lângă informația propriu-zisă pe care o transmit, fie prin ele însele fie completând informațiile text, trebuie să asigure și un cadru plăcut, sugestiv pentru receptarea mesajelor. În această direcție, realismul vizual al reprezentărilor grafice joacă un rol de primă importanță indiferent dacă grafica este utilizată în aplicații pentru comunicația cu utilizatorul sau dacă este obiectul principal al aplicației (animatie, grafica de prezentare, jocuri, etc.).

Obținerea realismului vizual al reprezentărilor grafice este un proces în general complex, care solicită resurse hardware și software considerabile. Timpul de calcul implicat în acest proces este uneori deosebit de mare ceea ce face ca, atunci când aplicația este sensibilă din punctul de vedere al timpului de elaborare al imaginilor grafice, acest proces să fie întrerupt în diverse faze ale sale. În aceste condiții calitatea reprezentărilor este desigur afectată.

2. Reproducerea umbrelor în contextul realismului vizual

Etapile principale parcurse în procesul obținerii realismului vizual corespund unor prelucrări grafice de complexitate din ce în ce mai mare. Vom considera ca obiectele componente ale unei scene grafice sînt modelate prin rețele poligonale.

Prima etapă cuprinde procesul de îndepărtare al liniilor și suprafețelor ascunse pentru obiectele scenei grafice 3D. Atunci cînd un obiect este observat dintr-un anumit punct în spațiu, anumite fațete ale sale sînt vizibile complet, altele ascunse parțial sau total. Îndepărtarea suprafețelor ascunse este o necesitate atît în procesul obținerii realismului vizual al imaginilor generate cit și pentru reducerea drastică a efortului

de prelucrare solicitat, în etapele următoare.

Etapa următoare cuprinde ansamblul de prelucrări orientate pe redarea iluminării scenei grafice. Reprezentarea finală a suprafețelor depinde de tipurile surselor de lumină amplasate în scenă, de proprietățile fizice ale suprafețelor (culoare, textură, reflectivitate) precum și de pozițiile și orientările suprafețelor în raport cu sursele de lumină.

O nouă etapă calitativ superioară este reproducerea proprietăților suprafețelor obiectelor. În această etapă sunt aplicate tehnici specifice de texturare și nuanțare a culorilor, obținându-se efecte vizuale deosebite.

Un grad sporit de realism vizual se obține prin reproducerea umbrelor aruncate de obiectele iluminate asupra celorlalte obiecte din scena grafică.

În cadrul lucrării se prezintă o posibilă implementare într-un pachet grafic elaborat în limbaj evoluat (Pascal) a metodei de reproducere a umbrelor având la bază o variantă a tehnicii poligoanelor detaliu asociate, și integrarea acesteia în fluxul prelucrărilor grafice care conduc la reprezentarea scenelor grafice 3D.

3. Tehnica poligoanelor detaliu

Tehnica poligoanelor detaliu se bazează pe observatia că suprafețele care sînt în umbră sînt ascunse observatorului atunci cînd scena este privită din punctul de amplasare al unei surse de lumină. Pe baza acestei observații, procesul de determinare al umbrelor este similar cu procesul de identificare al suprafețelor ascunse.

Tehnica poligoanelor detaliu poate fi aplicată cu succes în mod deosebit în cazul în care modelarea poligonală este combinată cu o metoda de transpunere pe dispozitivul grafic de ieșire raster bazată pe sortarea în adîncime și pe algoritmul de acoperire painter.

Determinarea umbrelor ca etapă de procesare, poate fi făcută în paralel cu procesul de identificare al suprafețelor ascunse. Algoritmul de identificare al suprafețelor ascunse stabilește care sînt suprafețele vizibile din punctul curent de observare, în timp ce algoritmul de determinare a umbrelor stabilește care sînt suprafețele "vizibile" din punctele în care sînt amplasate sursele de lumină.

Esenta procesului de determinare al umbrelor este observatia că acele suprafețe poligonale care sînt vizibile atît din punctul de observare cît și din punctul de amplasare al sursei nu pot fi în umbră iar acelea care sînt vizibile din punctul de observare dar nu și din punctul de amplasare al sursei sînt în umbră. Aceasta logică poate fi extinsă pentru surse punctuale multiple[1].

În principiu, fiecărui poligon vizibil din punctul de observare care este complet sau parțial vizibil din punctul de amplasare al sursei (este "în lumină") i se atasează un poligon secundar copiat și numit poligon detaliu. Umbra aruncată de un poligon P pe un poligon Q este un nou poligon U. Pentru un

poligon P se determina reuniunea poligoanelor umbra determinate de celelalte poligoane ale scenei care sînt în lumina de la cel puțin o sursă. Acest poligon rezultat R este poligonul detaliu asociat poligonului P . Poligonul P se transpune în ecran utilizînd iluminarea calculată pentru acesta iar apoi se transpune poligonul R (peste P) luînd în calculul de iluminare numai intensitatea luminii ambiante.

4. Implementarea tehnicii poligoanelor detaliu

Vom considera că în scena grafică sînt prezente N surse de lumină punctiforme, o iluminare ambientală și un număr oarecare de corpuri solide descrise prin rețele de poligoane nonpenetrante care nu se acopera în mod ciclic.

Ansamblul de poligoane care descriu obiectele prezente în scena grafică este supus procesului de identificare al suprafețelor ascunse în paralel cu procesul de determinare a vizibilității din cele N puncte de amplasare ale surselor de lumină. Cele două procese fiind în esență aceleași, se utilizează aceeași procedură bazată pe evaluarea cosinusului unghiului făcut de normală la suprafața poligonului (orientat) și direcția de observare respectiv direcțiile date de amplasarea surselor. În structura de date atașată unui poligon sînt marcate rezultatele procesării; dacă este ascuns sau nu și numerele de identificare ale surselor fata de care poligonul este în lumină.

În etapa următoare poligoanele sînt sortate în funcție de departarea centrului de greutate al poligonului fata de poziția de observare. Se obține o listă avînd NRP poligoane sortate care corespunde cu ordinea în care poligoanele vor fi transpuse în ecran. Primul poligon din listă este poligonul cel mai îndepărtat de poziția de observare.

Datorită ordonării poligoanelor, pe un poligon P_i vizibil din punctul de observare pot apare în principiu umbre aruncate de oricare din poligoanele P_j ($i < j \leq NRP$) care nu au fost încă transpuse în ecran și care sînt în lumină relativ la cel puțin una din sursele de lumină. Poligoanele P_j ($j < i$) care se găsesc în spatele poligonului P_i nu sînt luate în considerare. Umbrele aruncate de acestea vor apare, eventual, pe fata poligonului P_i care nu este vizibilă din punctul de observare.

Dupa transpunerea în ecran a poligonului P_i , se determină cite un poligon detaliu $P_{j,i}$ generat de fiecare poligon P_j , $i < j \leq NRP$, care este în lumină fata de cel puțin una din cele N surse de lumină. Acest poligon obținut se transpune în ecran luînd în calculul de iluminare doar intensitatea luminii ambientale.

Determinarea unui poligon $P_{j,i}$ (poligon detaliu generat de poligonul P_j relativ la poligonul P_i) presupune următoarele operații:

- se determina ecuația planului poligonului P_i
- se realizează o proiecție paralelă a poligonul P_j pe planul poligonului P_i , considerînd o iluminare paralelă din direcția sursei K_i obținînd poligonul detaliu $P_{j,i}$
- se determină extinderile pe ox respectiv oy ale

poligoanelor P_i și P_{j+1} proiectate în spațiul ecranului. Si se face un test de suprapunere al extinderilor. Dacă cele două poligoane nu se suprapun se trece la procesarea următorului poligon din listă.

-dacă poligoanele se suprapun, poligonul P_{j+1} se supune unui proces de clipping pentru poligoane (de exemplu Sutherland-Hodgman), considerind ca regiune de clipping poligonul P_i . Se obține poligonul sectionat Q_{j+1} .

-dacă poligonul Q_{j+1} există, se transpune în ecran luând în calculul de iluminare intensitatea luminii ambientale

4. Concluzii

Metoda prezentată produce reprezentări grafice cu grad ridicat de realism vizual. Este recomandată atunci când scenele grafice sînt de complexitate medie, timpul de execuție crescînd substanțial atunci cînd numărul poligoanelor din scenă grafică este foarte mare.

Bibliografie

1. Van Dam - Principles of Interactive Computer Graphics
2. Rankin - Computer Graphics Software Construction
3. - Mathematical Elements for Computer Graphics

Metoda de recunoaștere a obiectelor poligonale

Silviu Iftin

Facultatea de Automatică și Calculatoare

Institutul Politehnic Iași

Rezumat

Această lucrare prezintă o metodă eficientă de recunoaștere a obiectelor poligonale supuse la translație, rotație și scalare cu un factor global din imagini alb-negru. Metoda de recunoaștere se bazează pe găsirea corespondenței între forma analizată și o bibliotecă de modele anterior creată. Compararea cu modelele din bibliotecă se face în două treceri în fiecare din acestea utilizând un anumit tip de trasături.

1. Introducere

Tehnicile de clasificare a formelor bazate pe contur au constituit o importantă zonă de cercetare în ultimii ani și au cuprins metode statistice, analiza Fourier a unor funcții dependente de contur, metode sintactice etc. Recunoașterea unui obiect implică realizarea unei corespondențe între o apariție particulară a acestuia și un model memorat anterior într-o bibliotecă de obiecte cunoscute. Această corespondență se realizează prin compararea unor caracteristici. Ordinea în care se determină valorile caracteristicilor în model și într-o apariție particulară depinde însă de poziția în care sînt considerate aceste obiecte. Este necesară deci folosirea unor algoritmi de aliniere a componentelor celor doi vectori sau a unor etape pregătitoare de căutare sau indexare a căror complexitate este mare (în general $O(n^3)$) [2]. Cum dimensiunea acestor vectori și a numărului de modele este în general mare timpul necesar recunoașterii crește simțitor.

În această lucrare pentru recunoașterea obiectelor poligonale s-au ales două tipuri de trasături și anume unghiurile formate de laturile poligonului și lungimile acestor laturi. Este evident că primul tip este invariant la translație, rotație și scalare în timp ce al doilea este invariant la primele două operații și liniar în raport cu cea de a treia [4]. Aceste două tipuri de trasături asigură totodată un plus de acuratețe în recunoaștere și se pot calcula ușor. Nici pentru aceste trasături însă nu se poate stabili un punct de start și o ordine de calcul invariabilă în raport cu poziția obiectului. Pentru a rezolva această cerință de aliniere s-au folosit în comparare vectorii prelucrați pe baza unei transformări descrise în continuare.

2. Descrierea transformării Rapide

Transformata Rapida realizează o corespondență între coeficienții unui vector de intrare și un vector de ieșire avind proprietatea ca vectorul de ieșire este invariant la deplasări ciclice ale componentelor vectorului de intrare (1). Considerăm un vector $V = (a_0, \dots, a_{n-1})$ unde n este de forma 2^m . Operatorii folosiți în transformare sînt suma (+) și modulul diferentiei [-] fiind posibilă eventual și o altă alegere. Definirea transformatei se poate face utilizînd următoarele formule recursive :

$$(1) \quad R(a_0, \dots, a_{n-1}) = (T^0(a_0), \dots, T^0(a_{n-1})) \quad \text{unde :}$$

$$(2) \quad \begin{cases} T^0(a_i) = a_i & \forall i = 1, 2, \dots, n \\ T^k(a_{2i-1}) = T^{k-1}(a_i) + T^{k-1}(a_{i+n/2}) & \forall i = 1, 2, \dots, n/2 \\ T^k(a_{2i}) = [T^{k-1}(a_i) - T^{k-1}(a_{i+n/2})] & \forall i = 1, 2, \dots, n/2 \end{cases}$$

Se poate demonstra că transformata astfel definită este liniară și invariantă la deplasări ciclice ale componentelor vectorului de intrare. Calcularea ei se face utilizînd operații simple iar complexitatea algoritmului de calcul este $O(n \log n)$. În ipoteza păstrării aceluiași sens de efectuare a transformărilor pentru cele două trăsături aplicarea transformatei vectorului real și a celui de test determină alinierea componentelor :

Un inconvenient al acestui procedeu îl constituie faptul că numărul de operații din vector trebuie să fie o putere a lui 2 ceea ce este dificil în practică. O metodă de evitare a acestui dezavantaj folosită și în calculare este efectuarea cîte completări numărului de componente ale vectorului prin adăugarea de zerouri astfel : dacă $2^{m-1} < n < 2^m$ atunci se completează primale $2^m - n$ componente în ordinea mărimii lor și se inserează un zero după fiecare din ele.

3. Algoritmii de recursivitate

Considerăm un poligon n-arecun, P cu n vîrfuri aștut într-un reper cartezian (x, y) și poligonul obținut în urma operațiilor de scalare cu un factor de scală global ($x = x' \cdot a$, $y = y' \cdot a$) și apoi rotit și traslat.

Este evident că vîrfulurile celor două poligoane sînt două cîte două egale iar laturile două cîte două proporționale. Prima etapă a algoritmului creează biblioteca de obiecte sub forma unei colecții de înregistrări cuprîndînd

numărul de vîrfuri, vectorul conținînd mărimea unghiurilor și cel care conține mărimea laturilor după aplicarea transformării descrise anterior.

Faza de clasificare propriu-zisă pornește cu un algoritmul de depistare, calculare a coordonatelor vîrfurilor bazat pe detecția segmentelor de dreaptă prin metoda transformei Hough [5] urmată de determinarea celor doi vectori utilizați anterior -unghiuri, laturi- și a transformei lor. Cautarea colecției create se face pe baza numărului de vîrfuri și a distanței între vectorii transformați cuprinzînd mărimea unghiurilor. În cazul cînd această distanță se găsește sub un anumit prag se selectează intrarea corespunzătoare din bibliotecă și se determină o medie a coeficientului de proporționalitate laturilor în raport cu vectorul calculat pentru poligonul de recunoscut:

$$(3) \quad \alpha_{\text{med}} = \frac{\sum_{i=1}^n T^m(\ell'_i)}{\sum_{i=1}^n T^m(\ell_i)}$$

unde ℓ_i și ℓ'_i sînt lungimile laturilor celor două poligoane. Se are în vedere că transformata este liniară deci:

$$(4) \quad T(\alpha \cdot a) = \alpha \cdot T(a)$$

Se verifică dacă acest coeficient satisface cu o anumită aproximație δ , aleasă convenabil, proporționalitatea fiecărei componente pentru vectorii în bibliotecă selectați în prima fază și se rețin aceia pentru care este satisfăcută relația (5):

$$(5) \quad \left| \alpha_{\text{med}} - \frac{T^m(\ell'_i)}{T^m(\ell_i)} \right| < \delta \quad \forall i = 1, 2, \dots, n$$

Se reia operația de comparare considerînd vectorul lungimilor scalat cu coeficientul găsit și decizia finală se ia în funcție de distanța vectorilor transformați corespunzători.

Întrucît în evaluarea acestor distanțe sau folosit vectorii transformați nu este necesară conform proprietății transformei Rapide păstrarea aceleiași ordini de înregistrare în vector a componentelor sau a unei alinierei ulterioare.

4. Concluzii

Rutinele algoritmului prezentat au fost implementate și testate pe un calculator compatibil IBM-PC în limbajul Turbo-C folosind imagini alb-negru cu o rezoluție de 256 X 256 puncte. Metoda poate fi extinsă la curbe carecare având în vedere posibilitatea aproximării acestora prin linii poligonale și alegând o altă trăsătură în locul unghiului dintre segmente având în vedere posibilitățile diferite de poziționare a virfurilor. O modalitate de a crește robustețea acestui algoritma constă în eliminarea unor virfuri corespunzătoare unor unghiuri mari (aproape de 180°) care pot ascunde erori în faza de preprocesare.

Bibliografie

1. H.A. Al-Mohamed - "3D Shape Classification using the R- transform". 10 th International Conference on Pattern Recognition, IEEE Computer Society Press 1990.
2. Fridtjof Stein, Gerard Medioni - "Efficient Two Dimensional Object Recognition", 10 th International Conference on Pattern Recognition, IEEE Computer Society Press 1990.
3. Ilan Schrouber, Moshe Ben-Bassat - "Polygonal Object Recognition". 10 th International Conference on Pattern Recognition, IEEE Computer Society Press 1990.
4. Dorian Dogaru - "Metode noi în proiectare. Elemente de grafică tridimensională", Ed. Științifică și Enciclopedică, București 1988.
5. D.H. Ballard, C.Brown - "Computer Vision", Prentice Hall 1982.

SISTEM DE LABORATOR PENTRU DEZVOLTAREA APLICATIILOR DE CONDUCERE IBM-PC SPOT

Autori: Dr. ing. D. Carstoiu
Ing. C. Radu
Ing. M. Litoiu
Institutul Politehnic Bucuresti

1. INTRODUCERE.

Lucrarea de fata propune constructia cu echipamente existente in tara a unei aplicatii de laborator, destinata in principal instruirii in domeniul conducerii proceselor. Structura propusa este compusa dintr-un echipament SPOT, avind ca scop exploatarea particularitatilor sale legate de existenta plachetelor de interfata pentru cuplare la proces, si un calculator IBM PC (sau compatibil) pentru facilitatile legate de viteza de prelucrare cit si avantajelor oferite pentru realizarea consolei operatorului de proces.

Cele doua echipamente sînt conectate printr-o legatura seriala cu suport de comunicatie între echipamentul SPOT si calculatorul IBM PC.

Din punct de vedere software s-au realizat pachetele de programe pentru achizitie de date, prelucrare primara, transmisie comenzi, programele de comunicare scrise in CAZTEC la echipamentul SPOT. La calculatorul IBM PC pe lînga programele de comunicatie s-a realizat un pachet de programe de consola in C++ in care s-au folosit concepte ale programarii orientate obiect.

2. ARHITECTURA HARDWARE A SISTEMULUI

Stiut fiind faptul ca un calculator IBM PC nu este realizat in varianta industrială pentru a permite utilizarea sa in medii perturbate prin cîmp magnetic si electric, precum si in medii cu praf si agenti corozivi se incearca utilizarea acestuia si in aplicatii de conducere datorita atit a puterii de calcul cit si a gamei mari de programe realizate sub sistenul de operare DOS.

Echipamentul MINISPOT are o interfata seriala de tip SIO, microcalculatoarele din familia IBM PC, XT, AT folosesc pentru realizarea interfetei seriale circuitul 8250 UART (universal asynchronous receiver transmitter). Sistemul de operare DOS suporta doua porturi seriale (recunoscute de sistemul de operare sub numele de COM1 si COM2), si deci doua circuite UART. Adresele de baza ale celor doua porturi sînt tinute la 0040:0000 pentru COM1 si la 0040:0002 pentru COM2. Adresa de baza pentru COM1 este 3F8H iar pentru COM2 este 2F8H. Pentru simplificare, referirile se vor face la registrele numerotate 3F8H dar sînt valabile si pentru registrele de la 2F8H.

Circuitul 8250 are 10 registre de 8 biti selectate ca sapte porturi cu adrese distincte, 3FBH-3FEH. In cinci cazuri, registrele sint accesate la o adresa particulara de port, in functie de valcarea bitului 7 din registrul de control al liniei. In continuare sint prezentate registrele :

3FBH(OUT ,bit 7=0 la 3FBH)	Registru de transmisie
3FBH(IN ,bit 7=0 la 3FBH)	Registru receptie data
3FBH(OUT ,bit 7=1 la 3FBH)	Baud-rate low
3F9H(OUT ,bit 7=1 la 3FBH)	Baud-rate high
3F9H(OUT ,bit 7=0 la 3FBH)	Registru de activare intrerupere
3FAH(IN)	Registru de identificare a intreruperii
3FBH(OUT)	Registru de control al liniei
3FCH(OUT)	Registru de control al modemului
3FDH(IN)	Registru de stare a liniei
3FEH(IN)	Registru de stare a modemului

3. STRUCTURA SISTEMULUI DE PROGRAME

Sistemul de operare realizeaza initializare prin INT 14H, functia 0. In DX se incarca numarul canalului de comunicatie (0 pentru COM1 si 1 pentru COM2). In AL se incarca octetul de comanda ce contine lungimea cuvintului, numarul bitilor de stop, tipul paritatii si viteza de transfer.

Se da mai jos un exemplu de initializare a canalului 1 de comunicatie seriala cu toatele caracteristici: lungimea de cuvintului de 8 biti, 2 b. de stop, paritate impara, rata de transmitie 1200 bps:

MOV AL,0	;sterge al
OR AL,00000011B	;lungime cuvint
OR AL,00000000B	;biti de stop
OR AL,00011000B	;paritate
OR AL,10000000B	;rata de transmisie
MOV AH,0	;functia 0
MOV DX,0	;COM1
INT 14H	

Programatorul poate actiona direct asupra registrelor circuitului 8250, functie de semnificatia bitilor registrului de control al liniei. In continuare se prezinta listingul scris in limbaj C al programelor de comunicatie implementate pe IBM PC.

```

/*****
functii de comunicare :
    transmit date
    receptioneaza data
*****/

#include (stdio.h)
#include (string.h)
#include (dos.h)

```



```

#define EOADR 0x32
#define BOADR 0x31
#define BOVAL 0x33
#define NRCAN 24
//structura pentru convertor
typedef struct{
    unsigned char vallow;    //LSB convertor
    unsigned char valhigh;   //MSB convertor
    char adr_can;           //adresa canale MUX
    char activ;             // canal ativ/inactiv
}pach;
pach par_CAN[NRCAN];

void prog_8250() //programare 8250
{
    asm(
        mov al,0
        or al,11b //lungime cuvint:8biti
        or al,100b //2 biti de stop
        or al,11000000b //viteza 4800
        mov ah,0
        mov dx,0 //COM1
        INT 14h
        mov al,0
        or al,11b //lungime cuvint:8biti
        or al,100b //2 biti de stop
        or al,11000000b //viteza 4800
        mov ah,0
        mov dx,1
        INT 14h //COM2
    )
}

void trans_char( char c)//scriere un caracter la COM1
{
    asm(
        mov ah,1 //functie 1 de la INT 14
        mov dx,0 //COM1
        mov al,c
        INT 14h
        /* mov stcom1,ah
        mov ah,1 //functie 1 de la INT 14
        mov dx,1 //COM2
        mov al,c
        INT 14h
        mov stcom2,ah */
    )
}

unsigned char rec_char()//receptie un caracter
unsigned char st_rec1;
unsigned char car_rec;
st_rec1=0x80;
while (st_rec1 != 0x80 )
{
    asm(

```



```

145
mov ah,2
mov dx,0
INT 14h
mov car_rec,al
mov st_rec1,ah
}
return(car_rec);
}
void rec_str(char *string){
char c[2]="a";
while((c[0]=rec_char())!=EOF)&&(strlen(string)<=30){
strcat(string,c);
}
}
void trans_adr(void){ //transmitere adresa pentru canale active
// adresele sint delimitate de BOADR si
//EOADR
char i;
trans_char(BOADR);
for (i=0;i(NRCAN;i++){
if (par_CAN[i].activ) trans_char(par_CAN[i].adr_can);
trans_char(EOADR);
}
}
void rec_valhex(void){//receptie valori hexa achizitionate de
// MINISPOT
char i,j;
trans_char(SOVAL); //initiere receptie
for (i=0;i(NRCAN;i++){
if (par_CAN[i].activ){
par_CAN[i].valow=rec_char();
par_CAN[i].valhigh=rec_char();
}
}
}
void main()
{
int i;
inportb(0x2FB);
for (i=0;i(NRCAN;i++){
par_CAN[i].valow=0;
par_CAN[i].valhigh=0;
par_CAN[i].adr_can=i;par_CAN[i].activ=1;
}
par_CAN[5].activ=0;
prog_8250();
trans_adr();
printf("continue:");
getch();
while (!kbhit()){
rec_valhex();
for(i=0;i(NRCAN;i++){
printf("canal %d %x %x\n",i,
par_CAN[i].valow,par_CAN[i].valhigh);
}
trans_char('\n');
}
}

```


Un program similar este implementat pe echipamentul SPOT, avind urmatoarele functii:

1. O parte de initializare in care se receptioneaza numarul canalelor MUX pe care se face achizitia;
2. realizeaza achizitia pe canalele solicitate;
3. trimite valorile achizitionate pe interfata seriala S2 si reia programul de la pasul 2.

```
#define BOADR 0x31
#define EQADR 0x32
#define CANPORT1 0x21
#define CANPORT2 0x22
#define CANPORT3 0x23
#define CONVNOTRDY 0x40
#define SOVAL 0x33
struct {
    char vallow;
    char valhigh;
    char adr;
    char activ;
} par_can[24];

main()
{
    char cmd;
    int i;
    prog1();
    actmag();
    /* initializari */
    for(i=0;i(24;i++)
    {
        par_can[i].vallow=0x01;
        par_can[i].valhigh=0x04;
        par_can[i].adr=i;
        par_can[i].activ=0;
    }
    /* se asteapta primirea areselor canalelor active */
    while( rec_c() != BOADR);
    rec_a();
    /* se comunica operatorului ce s-a receptionat */
    for(i=0;i(24;i++)
        printf("%d",par_can[i].activ);
    /* se repeta pasii 1 si 3 din algoritmul pina se primeste de la
    MASTER un caracter SOVAL */
    while ( rec_c() == SOVAL){
        achiz();
        for (i=0;i(24;i++) printf("canal %d %4 %4\n",
            i,par_can[i].vallow,par_can[i].valhigh);
        tr_val();
    }
}

Nu se prezinta in aceasta lucrare din considerente de
```

parte din funcțiile apelate cum sînt: prog1() (initializarea legăturii seriale), rec_a() (recepția adresei canal analogic impusă de PC), rec_c() (recepția unui caracter pe legătura serială), tr_val() (transmisia valorii colectate de MINISPOT la PC). Pentru exemplificarea achiziției de date la nivelul echipamentului MINISPOT sînt date mai jos funcțiile C asociate.

```
achiz1() { /*achiziție canale active*/
int i;
for(i=0;i<(24;i++){
    if (par_can[i].activ) {
        out(CANPORT1,par_can[i].adr);
        while((in(CANPORT1)&CONVNOTRDY));
        par_can[i].valow=in(CANPORT3);
        par_can[i].valhigh=in(CANPORT2);
    }
}

actmag() { /*activare magistrala proces*/
    out(0xc0,0x01);
    out(0xf1,0xf0);
}
```

În continuare se exemplifică modul de construcție a consolei operatorului de proces la nivelul PC, utilizînd concepte ale programării orientate obiect. Elementele de interfață utilizate în proiectarea consolei operatorului de proces sînt cele utilizate în consolele tradiționale: butoane, comutatoare, înregistratoare (osciloscopae), afișări digitale, stări contacte (bec aprins/bec stins).

Realizînd o ierarhizare a elementelor de interfață obținem structura din figura 1. La baza ierarhiei se află clasa aparat înzestrată cu funcțiile din dreapta (m - desemnează o metodă). Aceste funcții sînt necesare pentru proiectarea interactivă a consolei. Pe următorul nivel al ierarhiei se află două clase: clasa aparate de afișare și clasa aparate de comandă. Aceste două clase moștenesc elementele clasei de bază dar posedă și alte elemente ce le individualizează. Pe nivelul 3 al ierarhiei este rafinat nivelul 2.

Aparatele trebuie construite astfel încît să permită o proiectare a consolei. De aceea ele sînt dotate cu metode pentru poziționarea lor pe ecran și pentru stabilirea dimensiunii lor. Prescurtarile au următoarea semnificație: ls - limita superioară, li - limita inferioară, vr - valoarea reală de afișat, rata - perioada de esanționare, buffer - zona de memorie pentru stocarea cîtoror esanționate.

Lucrînd cu obiecte de genul celor prezentate proiectarea programului se apropie foarte mult de proiectarea schemelor electrice ceea ce este un avantaj deosebit. În plus obiectele construite devin instrumente software versatile.

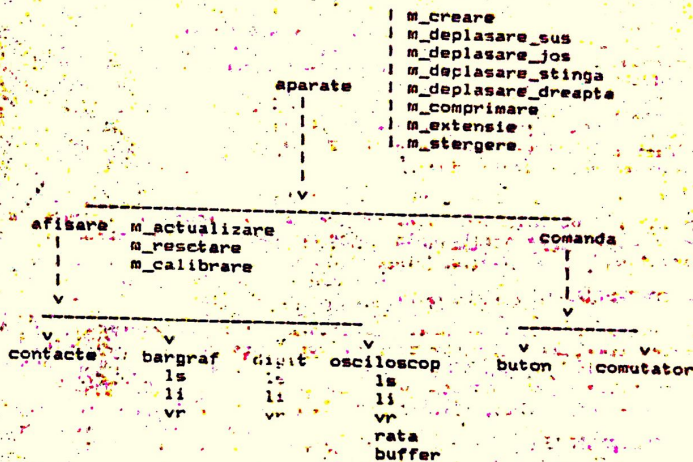


Fig. 1

Pentru implementarea în C++ s-a considerat ca la baza ierarhiei se afla clasa aparat cu următoarele elemente:
 -date de tip protected (protejate) care pot fi folosite de metodele clasei dar și de metodele claselor derivate;
 -metode definite cu cheia virtual. Pentru fiecare clasa derivată, metodele sunt implementate diferit, punându-se în evidență proprietatea de polimorfism.

Fisierul care conține clasa aparat cu numele AP.H este prezentat mai jos.

```

/*****
clasa aparat - sta la baza ierarhiei: se creeaza ce
se creeaza pentru consola;
*****/

class aparat
{
protected:
    float minvalue, maxvalue, sca, offset; //valoare maxima,
                                           //minima, scara,
                                           //offset
    char *title; //text
    float value; //valoare
    char *DSVSt;
    int LVMax;

```

```
int poly[8]; //coordonatele aparatului
int width,height; //latimea si inaltimea unui caracter
```

```
public:
//initializarea aparatului
virtual void Init(...) {}
//desenare aparat
virtual void Show(void){};
//stergere aparat
void virtual Hide(void){};
//actualizare valoare
virtual void Sample(){};
};
```

Delelalte obiecte nu sint prezentate in aceasta lucrare din considerente de spatiu, ele fiind implementate pe structura in discutie.

4. CONCLUZII

Structura propusa se poate constitui intr-un element initial pentru realizarea sistemelor de conducere pe doua nivele. In acelasi timp este un bun instrument de instruire atat pentru studenti cit si pentru alti beneficiari ce doresc sa se perfectioneze in domeniul sistemelor de conducere.

5. BIBLIOGRAFIE

1. IBM, Personal Computer Hardware Reference
2. P. L. Andersson, G. C. Anderson, Programming in ANSI C, 1989.
3. R. Lafore, Microsoft C Programming for the IBM, 1987.
4. Manual SPOT.
5. E. Eckel, Object - oriented programming with C++, 1990.

PACHET PENTRU ANALIZA PROGRAMELOR
DE AUTOMAT PROGRAMABIL PE BIT
(AP MICRO , AP 101 , AP 201)

as. ing. Calin Meza (I.P.Iași)
as. ing. Cornelii Căileanu (I.P.Iași)

1) Introducere

Automatele programabile (AP) sînt un element de bază în domeniul automatizărilor industriale (mașini-unelte , instalații din industria chimică sau metalurgică etc.) .

Setul relativ restrîns de instrucțiuni și caracterul relativ simplu al unor aplicații au dus uneori la desconsiderarea programării acestor echipamente .

Absența unor norme de lucru precise și stricte a dus la existența (pe utilajele aflate în exploatare) a unor programe cu deficiențe care merg de la necorelarea cu ansamblul documentației pînă la greșeli ce împiedică evident funcționarea conform prescripțiilor inițiale .

Pachetul de programe prezentat își propune analiza codului obiect ce ar conașta un automat programabil pe bit de fabricație românească (AP MICRO , AP 101-4K , AP 101-8K , AP 201) . S-a pornit de la prezenta posibilității existenței oricărei combinații de biți , fie și ca urmare a scrierii unui EPROM incomplet șters .

2) Descriere

2.1) Încărcarea codului obiect

Mu am dispus de un cititor de EPROM-uri așa că am proiectat în acest scop un dispozitiv conectabil la interfața paralelă de pe placa de bază a unui microcalculator Cub-Z .

Codul obiect este citit cu un program scris în "C" și în limbaj de asamblare și depus într-un fișier pe disc flexibil .

Fișierele astfel create au fost transferate la un minicalculator de tip INDEPENDENT sau la un calculator tip IBM-PC , pe care s-au făcut toate prelucrările ulterioare .

2.2) Analizarea programelor de AP

Toate programele din pachet sînt scrise în limbajul "C" și au ca rezultate finale fișiere text ce pot fi listate cu utilitățile sistemului de operare .

Principalii parametri ce trebuie indicați la lansarea programelor sînt tipul de AP (patru variante) , limitele zonei de inițializare și (eventual) opțiunea asupra părții analizate (programul utilizator sau de inițializare) .

Componențele principale ale pachetului de programe sînt :

(1) Dezasamblor

Programul prelucraază fișierul cu codul obiect de AP , rezultînd două fișiere listing : unul cu textul sursă și celălalt

cu adresele la care s-a detectat un cod incorect (care nu poate fi dezamplasat) .

Textul sursei este prezentat în pagini corespunzând la 50 de cuvinte de cod obiect, indicându-se adresa (în zeciazi), codul corespunzător (hexazecimal), instrucțiunea (aneonicul și operandul). La adresele cărora le corespund coduri incorecte apar în rubrica instrucțiunii semne de întrebare iar pentru adrese de salt în afara spațiului real permis se indică "JMP/JC ????".

(2) Program de listare exclusiv a instrucțiunilor de salt

Se creează un fișier listing ce conține numai instrucțiunile de salt din program, util în cazul folosirii intense a acestora sau ca bază de plecare în cazul necunoașterii tipului de AP pentru un cod obiect dat (încercînd toate cele patru alternative s-ar putea trage concluzii utile) .

(3) Listare referințe

Analizînd fișierul inițial (codul obiect), se creează un fișier cu variabilele folosite în program (intrări, ieșiri, memorii, temporizări - grupate în ordine crescătoare pentru fiecare tip) și adresele la care apar (tot în ordine crescătoare pentru fiecare variabilă) .

Nu apar în listingul generat variabilele nefolosite în program .

(4) Detectarea ieșirilor comandate condiționat

Programul creează un fișier listing în care sînt semnalate eventualele cazuri de ieșiri comandate prin instrucțiunile STO sau STOC în zone de program a căror parcurgere este condiționată de instrucțiuni JC .

(5) Analiza instrucțiunilor de salt

Se creează un fișier conținînd avertismente pentru următoarele situații : instrucțiuni de salt înapoi, salturi absurde (codul obiect corespunzător ar corespunde unui salt "pe loc"), sau eronate (pe al doilea cuvînt al unei instrucțiuni de salt - în AP 101-0X sau AP 201) .

(6) Analiza modului de utilizare a variabilelor

Din acest program rezultă un fișier intermediar în care se marchează modul în care sînt folosite (sau nu) variabilele I, M, T, E .

Se indică în comandă dacă se prelucrează programul utilizator sau cel de inițializare, precizîndu-se limitele zonei de inițializare .

(7) Semnalarea situațiilor de folosire dubioasă a variabilelor I, T, E

Pornind de la fișierul creat în programul (6), rezultă un listing în care se semnalează următoarele situații de utilizare potențial greșită a unei variabile :

- apare atât într-o instrucțiune STO cit și într-una STOC ;
- apare atât într-o instrucțiune STO/STOC cit și într-o S/R ;
- apare în mai multe instrucțiuni STO sau în mai multe STOC ;
- apare între instrucțiuni S dar nu și într-una R (sau

inversa);
 - apare într-o instrucțiune STO/STOC dar nu și în alta instrucțiune (diferită de S/R);
 - apare într-o altă instrucțiune dar nu și într-una STO/STOC/S (cu excepția variabilei T032 la AP MICRO).
 Nu se face automat nici o legătură între programul utilizator și cel de inițializare deci trebuie consultate și celelalte listingeri pentru eliminarea dubiilor.

(8) Crearea diagramelor de utilizare a variabilelor

Se pornește tot de la fișierul intermediar creat în (6), obținându-se în final un listing cu tabelele de gestionare a variabilelor I, M, T, E.

Diagramele sînt de forma unor tabele de cite 10 linii și 10 coloane pe pagină.

Pentru intrări se marchează dacă sînt folosite în program sau nu. Pentru memorii, temporizări și ieșiri se marchează modul de utilizare separat în instrucțiuni de tipul STO/STOC, S, R sau altele (blancuri pentru variabilele nefolosite).

Diagramele se încheie cu ultima variabilă (din fiecare categorie) folosită, în limitele logice specifice fiecărui tip de AP.

Evident, programele (7) și (8) se aplică separat pentru zona utilizator și cea de inițializare ale programului de AP.

9) Determinarea ecuațiilor sursă

Din analizarea codului obiect rezultă ecuațiile sursă corespunzătoare (cele care comandă efectiv mașina, nu neapărat cele dorite de proiectant sau de către producător).

Prin comandă se indică liniile programului de inițializare, care nu este luat în considerare (eventual poate fi tratat separat).

În urma prelucrării rezultă un listing cu ecuațiile sursă pentru toate instrucțiunile STO și STOC (indicându-se adresa instrucțiunii respective) și adresele pentru instrucțiunile S și respectiv R, aceasta pentru toate variabilele astfel comandate (M, T, E), în ordine crescătoare (M000, ..., M511, T000, ..., T511, E000, ..., E511).

O variantă a programului adaugă în listing și ecuațiile corespunzătoare instrucțiunilor S și R.

Se adăpt mai multe instrucțiuni de același tip (STO/STOC/S/R) pentru aceeași variabilă.

Nu se iau în considerare eventuale ecuații înălțuite prin instrucțiuni de salt. Dacă o ecuație nu începe cu o instrucțiune LD sau LDC (caz tipic: mai multe instrucțiuni STO succesive), linia text va avea la început "!!!!".

Pentru facilitarea urmăririi ecuațiilor (și ținând cont că operatorii nu au nici un fel de prioritate la această gamă de automate programabile) se folosesc în scriere paranteze, puse după cum urmează (nici o legătură cu parantezele din sintaxa limbajului de la AP 117):

- între o instrucțiune OR/ORC sau AND/ANDC și o instrucțiune XOR;
- între o instrucțiune OR/ORC sau XOR și o instrucțiune AND/ANDC.

3) Concluzii

Acest pachet de programe oferă o bază exactă și consistentă

pentru exploatarea unui autoser programabil pe bit, atât în faza de recepție a unei mașini sau instalații cît și la întreținerea ulterioară, sau pentru detectarea și eliminarea cauzelor ce duc la o funcționare incorectă și chiar la avarii grave.

Prin aplicarea întregului set unui program de AP se obține o documentație aproape completă (excepție făcînd, evident, sursa program) și exactă, eliminîndu-se neconcordanțele cauzate de erorile de scriere sau de neactualizarea documentațiilor.

De asemenea, forma de prezentare este foarte comod de urărit și se pretează la o multiplicare mai accesibilă decît cea uzuală (la heliograf).

În acest mod se reduce substanțial munca oboșitoare și sigaloasă pe care ar presupune-o efectuarea operațiunilor anuntate "manual", specialistul din branșă trecînd direct la analiza și corectarea deficiențelor.

La testări, timpul total necesar pentru aplicarea întregului lanț de programe a fost de circa 8-9 minute pentru un kilocuvînt de cod obiect.

Costul unei astfel de prelucrări ar fi deci de cîteva zecimi de procent din costul total al unui utilaj echipat cu un AP, net inferior pierderilor produse prin avarierea sau chiar numai staționarea acestuia.

Bibliografie

- 1) *** "AP 101. Manual de utilizare.", Automatica
- 2) *** "AP MICRO. Manual de utilizare.", Automatica

ANALIZA PROGRAMELOR PENTRU AUTORATELE PROGRAMABILE PE CUVINT (AP 117)

as.ing. Calia Neaga (I.P.Ingi)
as.ing. Cornelia Cailonea (I.P.Ingi)

1) Introducere

Autoatele programabile (AP) și-au cistigat un rol binecunoscut în domeniul automatizărilor industriale datorită avantajelor oferite (flexibilitate în proiectare , robustețe în exploatare etc.) .

După autoatele programabile pe bit au apărut cele programabile pe cuvint (la care se adaugă pe lângă setul de instrucțiuni secvențiale și operarea cu instrucțiuni numerice) , reprezentând un nivel de comandă superior .

Lipsa unor mijloace de proiectare asistată de calculator și a unor norme stricte a condus în unele cazuri la programe cu deficiențe mergînd de la neconcordanțe în documentație pînă la greșeli pe cit de obscure , pe atît de contraproductive .

Echipamentul de programare uzual (consola de programare CPP 286) , deși are un ansamblu mult mai riguros pentru AP pe cuvint decît pentru cele pe bit , oferă mijloace de control foarte limitate , înverate unui soft înscris în EPROM-uri .

Programele prezentate în continuare își propun analiza codului obiect ce ar comanda un AP pe cuvint de fabricație românească (AP 117) . Am păstrat principiile generale pe care se-aa bazat și în analiza programelor de AP pe bit .

2) Descriere

Pachetul de programe este scris în întregime în limbajul C și poate fi rulat pe un microcalculator de tip INDEPENDENT sau pe un compatibil IBM-PC . Rezultatele prelucrărilor sînt fișiere text ce pot fi listate cu utilizarea sistemului de operare .

Componentele principale sînt :

(1) Dezasmblor

Programul dezasmbliază codul obiect pentru AP 117 și creează două fișiere listing , unul cu programul sursă în pagini corespunzînd la cîte 50 de cuvinte de memorie și celălalt cu adresele la care s-a detectat un cod încorect (au poate fi dezasmblat corect sau este vorba de o greșeală de logică în folosirea operațiilor) .

O variantă a programului primește ca parametri în linia de comandă și adresele între care se face dezasmblarea .

(2) Program de listare exclusiv a instrucțiunilor de salt

Se analizează codul obiect pentru AP și se creează un fișier listing în care apar instrucțiunile de salt găsite în program ("JMP" , "JMPI" , "CAL" , "EXE" , "RPT") .

(3) Analizarea instrucțiunilor de salt

Se creează un fişier în care se semnalizează salturile dubioase (pe o instrucţiune "+", "n" etc.) sau incorrecte (pe unul din cuvintele unei instrucţiuni numerice altul decât primul).

De asemenea, se verifică adresele, indexul şi numărul de repetiţii pentru instrucţiunile "RPT".

(4) Urmărire folosire variabile

Programul creează un fişier binar intermediar în care se marchează modul de utilizare a variabilelor secvenţiale şi numerice.

Operaţiunea se face separat pentru programul de iniţializare şi pentru programul utilizator.

Se marchează şi variabilele care au apar explicit în programul sursă dar vor fi utilizate în execuţie prin includerea lor în bucle "RPT". Dacă prin indexări repetate s-ar depăşi spaţiul logic de adresare, se emite un mesaj de avertisment la terminal.

(5) Crearea diagramelor de utilizare a variabilelor

Folosind rezultatul de la (4), se obţine un fişier listing cu diagrame în pagini de câte 10 linii x 10 coloane pentru fiecare tip de variabilă în parte. Fiecare diagramă se încheie cu ultima variabila efectiv folosită în program.

Pentru variabilele secvenţiale se marchează următoarele variante de utilizare:

- într-o instrucţiune "=" sau "="/;
- într-o instrucţiune "S";
- într-o instrucţiune "R";
- în alta instrucţiune secvenţială;
- ca variabilă de lănsare;
- ca variabilă de marcat;
- pentru Carry sau Borrow;
- ca sursă sau destinaţie la o deplasare;
- într-un registru pas-cu-pas;
- folosită prin intermediul unei instrucţiuni "RPT".

Pentru variabilele numerice se indică următoarele posibilităţi de utilizare:

- ca sursă;
- ca destinaţie "RM/EN" sau numai "EN";
- sint alternat într-o instrucţiune;
- ca sursă pentru o instrucţiune "JMPI";
- ca destinaţie prin "ATV";
- folosită prin intermediul unei instrucţiuni "RPT".

La ieşirile numerice se semnalizează utilizarea unui canal special (care nu-şi modifică valoarea) sau a unui fictiv.

Diagrama prezintă rezultatele separat pentru programul utilizator şi pentru programul de iniţializare.

(6) Semnalarea cazurilor dubioase de gestionare a variabilelor

Folosind rezultatele de la (4) se obţine un listing în care se prezintă următoarele situaţii de eroare evidentă sau de dubiu:

- o intrare ar fi folosită ca destinaţie;
- o memorie, temporizare sau o ieşire secvenţială:
 - apare în mai multe instrucţiuni "=" sau "="/;
 - apare în "S" dar nu şi în "R";
 - apare în "R" dar nu şi în "S";

- apare ca destinație și nu este folosită ca sursă ;
 - apare ca sursă și nu este folosită ca destinație ;
 - o ieșire numerică ;
 - apare ca sursă ;
 - apare ca destinație în altă instrucțiune decât "TRAD" , "TRDD" , "TRDS" ;
 - se folosește un canal special sau unul fictiv .
- Mu se face nici o conexiune între programul utilizator și cel de inițializare deci trebuie corelate rezultatele cu celelalte listări .

(7) Listare referințe

Formind de la codul obiect pentru AP 117 , se creează un fișier cu referințele variabilelor (atât secvențiale cât și numerice) în ordinea lor crescătoare .

Mu apar în listing variabilele nefolosite în program ; nu se separă inițializarea de programul utilizator . În cazul folosirii într-o instrucțiune numerică , se indică adresa apariției efective , nu cea de început a instrucțiunii .

(8) Determinarea ecuațiilor (secvențiale) sursă

Prin acest program se creează un fișier cu ecuațiile secvențiale corespunzătoare instrucțiunilor "S" , "S/" , "S" și "R" , indicându-se și adresele lor . Se adăugă mai multe ecuații de același tip pentru aceeași destinație ; nu se separă programul utilizator de inițializare .

Destinațiile apar în listing în ordinea crescătoare (R000 , ... , R511 , T000 , ... , T511 , E000 , ... , E511) a celor comandate în program .

O variantă a programului indică pentru instrucțiunile "S" și "R" numai adresele la care apar .

Dacă o ecuație nu începe cu "!" , "!" sau cu deschiderea unei paranteze (deși asamblorul consolei de programare CPP 286 încearcă să excludă această posibilitate) , linia text va începe cu "!!!!" .

În scrierea ecuațiilor , parantezele sînt cele din sistaxa limbajului , deci trebuie să se țină cont de prioritatea operatorului "*" față de operatorul "+" . Dacă este cazul , la începutul ecuației se scriu numărul de paranteze deschise cerut de aritmetică și nu una singură , cum apare în programul sursă .

3) Concluzii

Creșterea complexității echipamentelor utilizate nu s-a corelat la noi cu o îmbunătățire substanțială a metodelor de lucru . Modul de lucru "manual" a dus în multe cazuri la scăpări ce dau multă bătaie de cap chiar numai pentru detectarea lor , remediile fiind extrem de dificile .

O simplă greșeală de scriere în documentație impune un efort serios chiar pentru o simplă comparație cu conținutul mesajilor EPROM cu care se lucrează , și aceasta cu condiția de a avea la îndemână o consolă de programare , lucru care nu se întâmplă întotdeauna .

Prin realizarea acestui pachet de programe ne-am propus să oferim o documentație exactă și consistentă pentru un program dat de AP pe cuvînt . Se elisaină în acest fel munca sigilantă și de uzură pe care ar presupune-o abordarea fără sprijinul calculatorului a detectării și analizării cauzelor ce conduc la o funcționare defectuoasă a unui utilaj echipat cu AP .

De asemenea, această documentație ar reprezenta un sprijin substanțial în activitatea curentă de întreținere.

Rezultatele acestor prelucrări permit să se treacă direct la analiza și corectarea deficiențelor.

Am avut la dispoziție pentru testare un program mai simplu dar putea estima timpul de calcul necesar aplicării întregului set de prelucrări la aproximativ 7-9 minute pe kilocuvint de cod obiect. Evident, costul unei astfel de prelucrări ar fi nesemnificativ în raport cu costul total al unui utilaj sau chiar numai cu cheltuielile de întreținere.

Bibliografie

- 1) sss "AP 117. Manual de utilizare.", Automatica.
- 2) sss "CPP 286. Manual de utilizare.", Automatica.

MODELAREA SI SIMULAREA CIRCUITELOR DIGITALE

Borgan Dorian, Dobricsean Sorin,
Rusu Dan, Ieremie Dana
Institutul Politehnic Cluj-Napoca

1. Introducere

Simularea si testarea schemelor logice corespunzatoare placilor echipate cu circuite digitale necesita construirea unui model. Construirea modelului se realizeaza interactiv prin tehnica meniurilor sau se poate utiliza un meniu realizat anterior pe baza informatiilor de structura extrase dintr-un fisier realizat cu mediul ORCAD. Modelul cuprinde informatii despre topologia placii, informatii legate de conectarea circuitelor, attribute ale circuitelor dintr-o biblioteca, valorile acestor attribute sau parametri de simulare si redactare a rezultatelor simularii.

Se prezinta o varianta de simulare orientata pe evenimente. Un eveniment se defineste ca fiind o modificare de stare logica a unui semnal intr-un nod de circuit. Gestionarea in timp a evenimentelor se face cu ajutorul unui secventiator. Secventiatorul determina evenimentul curent si creeaza o lista cu evenimentele urmatoare determinate de evenimentul curent. Simulatorul pastreaza o lista de evenimente posibile impreuna cu momentele producerii lor. El alege timpul de producere minim si executa actiunea declansata. Ca urmare lista de evenimente posibile se modifica in urma adaugarii de noi evenimente. Dupa care se revine la selectarea unui nou eveniment. Experimentul de simulare se termina la epuizarea timpului de simulare programat.

2. Modelarea sistemelor digitale

Topologia interna reprezinta imaginea plachetei externe in memoria calculatorului. Sunt reprezentate entitati ca si cai de legatura intre pini diferitelor componente de pe placheta, componentele fizice utilizate pe placheta impreuna cu legaturile dintre pini. Aceasta reprezinta structura fizica a plachetei. Topologia mai cuprinde o multime de attribute cu valorile lor, asociate pinilor: semnatura logica etalon, semnatura logica amprenta corespunzatoare diferitelor defecte utilizate in testare, valoare logica, nivel de tensiune, timp de propagare, fan-out maxim, etc.

In conceperea topologiei interne au stat la baza urmatoarele criterii:

- reprezentarea cit mai fidela a plachetei cu evitarea redundantei si inconsistentei datelor;
- posibilitatea accesului cit mai rapid la elementele din model, indiferent de sistemul de operare si limbajul de programare utilizat;

- posibilitatea modularizării topologiei și existența unor operații de partitionare, reuniune, salvare și încărcare în memorie a diferitelor module.

Topologia internă trebuie să asigure informațiile necesare parcurgerii plăchetei sau schemei logice după orice algoritmi pe același nivel ierarhic - toți pinii unei componente, toate caile din topologie, toți pinii unei componente tip sau fizice, toate atributele unui pin, etc; cit și între nivele ierarhice diferite - pin $\leftrightarrow$ componenta tip $\leftrightarrow$ componenta fizică, cale $\leftrightarrow$ pin $\leftrightarrow$ atribute, etc.

Topologia internă conține următoarele entități: cai (noduri de circuit), componente fizice de circuit (U1, Y23 utilizarea concretă a unui integrat digital), componente tip (CDB400, 74LS08, 8086, 4116), pinii componentelor tip (CLK, VCC, D, T, Q), pinii componentelor fizice (1, 14, 6), atributele și valorile acestora pentru pinii fizici și pinii tip (IN, OUT, TS, time, delay).

De exemplu, caile topologiei sunt descrise printr-o listă înlantuită de structuri, fiecare corespunzând unei cai. Fiecare structură conține informații de parcurgere a topologiei, informații necesare simulării cit și testării: nume cale, pointer la începutul listei ce constituie vectorul secventiatorului pentru respectiva cale, pointeri la expresiile Poage utilizate în determinarea vectorilor de test, starea de defect a nodului, numărul pinilor din cale, starea de eveniment a nodului la momentul curent de evaluare, pointer la lista pinilor din cale, pointer la următoarea cale din listă.

3. Construirea topologiei

Topologia se creează în fișiere text folosind programul NETLIST din produsul ORCAD cu opțiunea RACALREDAC și cu opțiunea /p pentru a se preciza pinii circuitelor prin număr și nu prin nume. Astfel se obțin două fișiere, unul cu extensia .CMP conținând asociațiile nume fizic componentă - nume tip componentă, și unul cu extensia .NET conținând conexiunile din topologie. Pentru generarea automată a programelor de test este necesar să fie precizate caile de intrare și ieșire din topologie (liniile la conector) și tipul circuitelor (analog sau digital) din topologie.

4. Secventiatorul

Secventiatorul conține vectorii de stare asociați fiecărui nod de circuit.

Notăm cu $vi[0]$ starea curentă a semnalului din nodul i la momentul curent t și cu $vi[1]$ starea următoare a semnalului din nod la momentul $t+T$, unde T este cuanta de timp de evaluare. Atunci $vi[dt]$ va fi starea la momentul $t+dt$. $vi[dt]$ se stabilește la momentul t evaluând funcția de transfer corespunzătoare pinului de ieșire care dictează valoarea logică a semnalului din nodul i . Pentru pinii de intrare care apar ca argumente pentru funcția logică, valorile lor se consideră ca valori $vi[0]$ din secventiator, fiind valorile semnalelor din

nodurile de intrare din care fac parte acesti pini.
Algoritmul de evaluare a semnalului dintr-un nod de circuit
si de determinare a unor evenimente viitoare este urmatorul:

```

evaluare(nod)
    daca eveniment(nod)
        pentru fiecare pin de intrare din nod
            $determina functia logica de transfer
            $evalueaza functia de transfer, folosind
            valorile curente din secventiator pentru
            toti pinii de intrare care apar ca argumente
            ale functiei de transfer
            $determina intirzierea corespunzatoare
            pinului de intrare curent
            $determina pinul de iesire care preia
            valoarea functiei de transfer
            $determina nodul din care face parte pinul de
            iesire
            $introduce in secventiator pe pozitia
            corespunzatoare intirzierii valoarea
            functiei de transfer
        altfel
            $nimic
    
```

Determinarea functiei de transfer se face relativ simplu:
-se determina componenta fizica din care face parte pinul de
intrare considerat;
-se determina componenta tip corespunzatoare componente
fizice;
-se determina numele functiei logice asociate componente
tip (NOT, AND3, OR2).

3. Simulatorul

Este un instrument foarte util in studiul functionarii unei
scheme logice digitale realizate cu circuite combinationale sau
secventiale de tip MSI. Este un simulator orientat pe eveniment,
are trei stari logice: 0 (zero logic), 1 (unu logic), X (stare
indiferenta), si starea de eveniment corespunzatoare celor doua
tranzitii: Up (tranzitie low → high), Down (tranzitie high →
low). Permite simularea a doua tipuri de defecte:

- nod de circuit blocat in 0 logic;
- nod de circuit blocat in 1 logic.

Simulatorul poate fi utilizat in doua moduri:

- a. Apelat direct. In acest mod se poate utiliza intregul set
de functii ale simulatorului, cum ar fi:
 - editarea interactiva a schemelor logice;
 - incarcarea, salvarea unei topologii editate;
 - editarea in diverse moduri a secventelor binare aplicate
la intrarea circuitelor simulate;
 - afisarea topologiei, bibliotecii de circuite, valorilor
logice a semnalelor din nodurile de circuit la fiecare pas de

simulare. Se pot edita in varianta actuala 40 de semnale de intrare pe durata a 256 pasi de simulare;

- programarea optiunilor de simulare, editare, afisare sau generare automata a vectorilor de test;

- simularea functionarii circuitului cu diverse optiuni programate;

- punerea in evidenta a comportarii circuitului la diferite frecvente de lucru, a aparitiei hazardului;

- generarea automata a vectorilor de test pentru defecte si puncte de vizualizare a defectelor programabile.

b. Apelat din preprocesor. In acest mod se utilizeaza facilitatea secventiatorului de a genera automat vectori de test pentru un circuit dat. Se incearca astfel gruparea automata a cit mai multe circuite intr-un acelasi modul de test. Rezulta o marire a vitezei de testare functionala si o reducere a numarului de puncte de test necesare intregii placi.

Lansarea directa a Simulatorului dintr-un meniu principal se face prin selectia comenzii Simulate. Dupa lansarea Simulatorului apare pe ecran meniul principal sub forma de bara orizontala. Cursorul luminos va fi positionat initial pe comanda Data File. Cu tastele sageti se selecteaza una din comenzi. Meniul bara contine urmatoarele comenzi: Data File, Topologie, Simulation, Options, Test si Quit. Pentru fiecare comanda selectata se afiseaza pe ecran un meniu specific. Comenzile principale au urmatoarele organizari ierarhice ale subcomenzilor:

Data File

1.EditData

- permite editarea datelor de intrare in simulator;
- 1.1.Up - afiseaza liniile 1-20 din matricea de datele de intrare;
- 1.2.Down - afiseaza liniile 21-40 din matricea de datele de intrare;
- 1.3.Left - afiseaza pagina precedenta paginii curente din matricea de date;
- 1.4.Right - afiseaza pagina urmatoare paginii curente din matricea de date;
- 1.5.Jump - salt la o pagina specificata;
- 1.6.Set - programarea unei linii de date de intrare cu o valoare logica 0 sau 1 specificata;
- 1.7.Randome - programarea cu valori aleatoare a unei linii de date pe un interval specificat;
- 1.8.Edit - se trece intr-un regim de editare a intregii matrici de date de intrare cu 0 sau 1. Deplasarea cursorului se face cu tastele sageti;
- 1.9.Quit - se revine in meniul principal al Simulatorului;

2.SaveData

- salveaza intr-un fisier cu numele specificat continutul matrici de date de intrare;

3.RetrieveData

- incarca dintr-un fisier specificat matricea cu datele de intrare;

Topologie

1. EditTopo - permite editarea interactiva de la consola a topologiei placii echipatei
 - 1.1. Place - insereaza o componenta fizica in topologie
Se specifica tipul si numele componentetei;
 - 1.2. Connect - conecteaza doi pini sau un pin cu o linie de date de intrare;
 - 1.3. Unconnect - desface legatura unui pin;
 - 1.4. Delete - contine o clasa de comenzi de stergere selectabile dintr-un submeniu
 - a. Component - sterge o componenta fizica specificata prin nume;
 - b. All - sterge toata shema logica;
 - 1.5. Show - afisari
 - a. Component - afiseaza o componenta precizata prin nume. Afiseaza attribute si conectariile pinilor;
 - b. Way - afiseaza o cale (nod de circuit) specificata prin nume. Afiseaza toti pinii din cale si attributele caii;
 - c. AllWays - afiseaza toate caile din topologie. Dupa afisarea unei cai se continua cu celelalte prin apasarea unei taste. Daca se tasteaza <ESC> afisarea se abandoneaza;
 - d. Library - afiseaza numele componentelor tip din topologie;
2. OrcadTopo - incarca o topologie dintr-un fisier de pe disc;
3. SaveTopo - salveaza o topologie din memorie pe disc;
4. SelectTopo - selecteaza topologia curenta cu care se lucreaza din multimea de topologii incarcate;
5. Partition - partitioneaza topologia al carei nume este cerut in module de test;

Module

1. Show - afisari referitoare la modul
 - a. Component - afiseaza lista de componente existente intr-un modul precizind atat numele fizic cit si tipul acestora;
 - b. Ways - afiseaza caile de intrare si caile de iesire dintr-un modul;
 - c. Pins - afiseaza caile si pinii circuitelor pe care au fost positionate cuile de test;

Simulation

- 1. Assign - asigneaza unui nod din schema logica o pozitie in afisarea formelor de unda din nodurile de circuit la fiecare pas de simulare;
- 2. ShowAssign - afiseaza asignarea curenta;
- 3. ClearAssign - sterge asignarea curenta;
- 4. Options - permite programarea unor optiuni referitoare la afisarea rezultatelor unui experiment de simulare;

4.1. FirstCycle

- programeaza primul pas de simulare care se afiseaza. Poate fi in domeniul 1...256 același cu adincimea matricii de date de intrare. Pe durata unui pas de simulare (Cycle) vectorul binar de la intrarea circuitului ramine neschimbat;

4.2. LastCycle

- programeaza ultimul pas de simulare care se afiseaza;

4.3. PagePause

- permite programarea pauzei intre ecranele ce afiseaza pasii de simulare;

4.4. WidthSubcycle

- programeaza latimea in pixeli la afisarea unui subciclu. Un subciclu se defineste ca fiind rata de evaluare a evenimentelor din nodurile de circuit;

4.5. Subcycle/Cycle

- programeaza latimea unui pas de simulare (Cycle) in numar de cuante de timp de evaluare;

4.6. MarkCycle

- marcheaza la afisare pasii de simulare;

4.7. MarkSubcycle

- marcheaza la afisare fiecare cuanta de timp de evaluare noduri de circuit;

Test

- genereaza automat vectorii de test pentru un modul a carui topologie si circuite se cunosc;

1. SetFault

- se precizeaza numele componentei fizice, numele pinului si valoarea defectului;

2. ClearFault

- anuleaza asignarea de la SetFault;

3. ShowFault

- afiseaza numele caii si tipul defectului;

4. SetVisiblePin

- se precizeaza numele componentei si pinul pentru care se asigura observarea defectului;

5. GenTestArray

- afiseaza vectorii de test;

Quit

- revenirea in meniul principal.

6. Fisierul de sabloane si de diagnoza

In urma generarii vectorilor de test rezulta fisierul de sabloane si de diagnoza. Fisierul de sabloane este un fisier binar, obtinut automat cu ajutorul Preprocesorului. Acest fisier are extensia implicita .WPL si contine urmatoarele informatii:

- numele formei de unda - nume simbolic asociat sablonului;
- numele componentei careia ii este asociat sablonul;
- numarul de semnale pe care se aplica sablonul;
- numele acestor semnale;
- numarul de secventa care se aplica circuitului;
- frecventa cu care se aplica secventele;
- secventele propriu-zise;

Fisierul de diagnoza este un fisier binar in care se specifica pasul din secventa de test si defectele care pot aparea in acel pas cu secventele binare care le pun in evidenta. Acest fisier are extensia implicita .DIA. Este creat automat de catre Preprocesor sau interactiv cu ajutorul editorului de sabloane.

7. Preprocesorul

Pentru modulele de test se face un test functional. Partitionarea placii in module de test se face automat de catre Preprocesor.

Pentru fiecare modul de test se determina:

- descrierea topologica cuprinzind circuitele digitale integrate ce compun modulul si lista de conexiuni dintre pini componentelor fizice;

- sabloanele de test si defectele puse in evidenta de catre fiecare sablon. Se creaza un fisier cu sablonul de test si un fisier de diagnoza. Fisierul de diagnoza precizeaza defectele care se pun in evidenta cu fiecare vector de test;

- lista nodurilor de circuit in care se vor amplasa cuie de test (driver sau senzor) din patul de cuie.

Partitionarea placii se face pentru circuitele a caror functionare se modeleaza in biblioteca de functii logice. In continuare se incearca extinderea unui modul cuprinzind si alte circuite de pe placa, dar in asa fel incit la bornele modulului sa se poata propaga orice defect singular din nodurile interne ale modulului.

Pentru determinarea vectorilor de test necesari pentru a pune in evidenta defectele din modul la pini de iesire ai modulului, se apeleaza Generatorul Vectorilor de Test din cadrul Simulatorului.

Vectorii de test, pentru fiecare modul, se memoreaza in fisiere cu sabloane. Pentru fiecare modul extins se creaza de asemenea, cite un fisier de diagnoza. In fisierul de diagnoza se inscriu raspunsurile de pe placa sub test corespunzatoare fiecarui defect din interiorul modulului.

8. Concluzii

Simulatorul de circuite poate fi utilizat cu succes în proiectarea asistată a schemelor digitale atât discrete cât și integrate. Variantele actuale a simulatorului urmează să fie îmbunătățite după verificarea mecanismelor de lucru cu semnale cu patru stări logice (0, 1, X și Z) și puteri diferite, două tranziții (Up și Down), sesizarea stărilor de hazard și a tranzițiilor simultane, a timpilor diferiți de propagare, a conflictelor în noduri, a defectelor simultane, etc.

9. Bibliografie

1. Vladutiu, M., Crisan M. - Tehnica testării echipamentelor automate de prelucrare a datelor, Editura FACLA, Timisoara 1989.
2. Breuer, M. A., Friedman, A. D. - Diagnostic and Reliable Design of Digital Systems, Computer Science Press Inc.
3. Chang, H. J., Manning, E., Metzger, G. - Fault Diagnosis of Digital Systems, Wiley-Interscience, New-York, 1970.

ASAMBLOR PENTRU SISTEME CU TRANSPUTER T414/T800

Autorii: Barsan Corneliiu
 Institutul Politehnic Iasi
 Albuliet Lucretia
 Bejenariu Silviu
 Roman Mariana
 Barsan Laura
 Institutul de Informatica Teoretica, Iasi

INTRODUCERE

Lucrarea prezinta un asamblor pentru transputerele T800 si T414 realizat la Institutul de Informatica Teoretica, Iasi. Exista in prezent mai multe sisteme cu transputere T414/T800 dezvoltate ca extensii pentru calculatoare IBM-PC sau ca sisteme independente. Aceste sisteme se livreaza impreuna cu medii de programare complexe incluzind compilatoare, asamblatoare, etc.

Asamblorul prezentat in lucrarea de fata face parte din categoria instrumentelor de dezvoltare si este destinat in principal proiectantilor de sisteme cu transputere.

In prezent, scrierea unui asamblor reprezinta o sarcina de complexitate medie, fara un caracter de nouitate deosebit. Totusi, anumite particularitati ale setului de instructiuni ale transputerelor fac ca problema generarii optime de cod sa nu fie chiar triviala.

Solutiile gasite pentru optimizarea codului sint, deasemenea, prezentate in lucrare.

1. TRANSPUTERELE T414/T800 - PREZENTARE GENERALA

Un transputer este un microprocesor RISC proiectat special pentru medii multiprocesor. Un chip (transputer) contine: un procesor, 2kb memorie RAM (4kb pentru T800), patru canale seriale DMA, doua timere si procesor de virgula flotanta (numai T800). Un executiv (cablat hard) asigura executarea proceselor specificate de doua liste de procese corespunzatoare a doua nivele de prioritate.

Setul de registre este format in principal din: trei registre generale utilizator organizate ca o stiva -A,B,C-, un registru operand O, un registru pointer de instructiuni I, un registru pointer spatiu de lucru W, doua perechi de registre pentru gestiunea listelor de procese.

Instructiunile sint de lungime fixa pe 8 biti: 4 biti pentru codul operatiei si 4 biti pentru operand. Setul instructiunilor

include 13 instrucțiuni de bază, două instrucțiuni de prefixare și o instrucțiune specială opr de indirectare.

Pregătirea în registrul operand a operandilor care nu pot fi reprezentați numai pe patru biti se face prin intermediul a două instrucțiuni pfix, nfix (pfix deplasează la stînga patru poziții registrul 0) după care operandul pe patru biti din instrucțiune este plasat în cîmpul 0-4 din 0.

O instrucțiune de forma operație <expresie> va fi tradusă de asamblor într-un număr corespunzător de instrucțiuni pfix sau nfix urmate de o instrucțiune operație.

Exemplu: Instrucțiunea J 100 se traduce în:

21	pfix 1
20	pfix 0
01	J 0

Completarea setului de instrucțiuni se asigură prin intermediul instrucțiunii operate precedată de o instrucțiune de prefixare. Instrucțiunea operate determină interpretarea conținutului 0 ca instrucțiune.

2. LIMBAJUL DE ASAMBLARE RECUNOSCUT, SINTAXA DE DESCRIERE

Asamblorul recunoaște limbajul specificat de firma INMOS [1] complementat cu o serie de directive de asamblare. Sintaxa liniei de program este următoarea:

[eticheta] delia operator delia [operand] [delia comentariu]

2.1 Cîmpul eticheta

Acest cîmp conține un sir de caractere alfanumerice împreună cu caracterul "-" din care primul caracter fiind obligatoriu o literă. Lungimea maximă a unei etichete este de maximum 40 de caractere. Literele mici se disting de cele mari.

O eticheta introduce un nume la care se asociază o valoare.

Această reprezintă, de regulă, valoarea contorului de jocat la linia respectivă. Excepție de la această regulă constituie liniile cu operatorul .eq, caz în care valoarea e data de cîmpul operand.

2.2 Cîmpul operator

Cîmpul operator pot apare:

- anemonice instrucțiuni
- nume directive
- nume macrouri

acest cîmp nu se face distincție între literele mari și

2.2.1 Mnemonice instructiuni

Sint acceptate mnemonicele din manualul MPKOS.

2.2.2 Numele directivelor

2.2.2.1. Directiva .align

Sintaxa : eticheta .align

Determina alinierea codului la o adresa para . Cind contorul de locatii contine o valoare impara se insereaza in cod un octet nul.

2.2.2.2. Directivele .db , .dw , .ds

Sintaxa : [eticheta] .db expresie
 [eticheta] .db "sir de caractere"
 [eticheta] .dw expresie
 [eticheta] .ds expresie

Directiva .db determina rezervarea unui octet de memorie, octet care se initializeaza cu valoarea data de cimpul operand.

Daca in cimpul operand apare o constanta de forma "sir de caractere" se genereaza un vector continind caracterele sirului.

Directiva .dw determina rezervarea unui cuvint (32 de biti) continind valoarea data de cimpul operand.

Directiva .ds se foloseste pentru rezervarea de zone de memorie. Lungimea in octeti a zonei rezervate e data de valoarea cimpului operand.

Cimpurile operand pentru directivele .dw, .db pot contine expresii evaluabile la prima sau a doua trecere , iar cimpul operand al directivei .ds trebuie obligatoriu sa contina o expresie evaluabila la prima trecere .

2.2.2.3. Directiva .eq

Sintaxa : nume .eq expresie

Permite definirea unui simbol si atribuirea unei valori simbolului respectiv. Numelui definit in cimpul eticheta i se asociaza valoarea expresiei din cimpul operand, expresie care trebuie sa fie evaluabila la prima trecere.

Simbolul definit cu directiva .eq poate fi folosit prin numele sau in expresiile din cimpurile operand.

2.2.2.4. Directivele `.list` și `.nolist`

Sintaxa : `.list`
`.nolist`

Determină activarea/dezactivarea generării fișierului `listing`.

2.2.2.5. Directiva `.org`

Sintaxa : `.org` expresie

Directiva `.org` schimbă valoarea contorului de locații cu valoarea dată de expresia operand, expresie care trebuie să fie evaluabilă la prima trecere.

Directiva `.org` nu poate fi folosită la rezervarea spațiului de memorie.

2.2.2.6. Directiva `.include`

Sintaxa : `.include` nume-fișier

Directiva `.include` are ca efect continuarea citirii de către asamblor la intrare a liniilor din fișierul specificat în "nume-fișier".

Fișierul care se include poate conține la rândul său directive `.include`, permitându-se maximum 6 nivele de adâncime.

2.2.2.7. Directivele `.if`, `.else` și `.endif`

Sintaxa : `.if` expresie
`...`
`corp instrucțiuni 1`
`...`
`.else`
`...`
`corp instrucțiuni 2`
`...`
`.endif`

sau

`.if` expresie
`...`
`corp instrucțiuni 1`
`...`
`.endif`

Directivele `.if`, `.else` și `.endif` implementează facilitatea de asamblare condiționată.

Dacă expresia a de valoare nenulă se assemblează liniile din "corp instrucțiuni 1". Dacă expresia are valoare nulă se assemblează liniile din "corp instrucțiuni 2".

2.2.2.8. Directivele .macro .mqls .label

Sintaxa :

```

.macro  nume-macro  arg1,arg2,...,argN
.label  et1,et2,...,etk
...
corp macro
...
.endm

```

Directiva .macro specifica numele unui macro si parametrii formali ai macro-ului. Linile din "corp macro" sint cele cuprinse intre directivele .macro si .endm.

Directiva .label declara etichetele locale din interiorul unui macro pentru care asamblorul va genera la fiecare invocare a macro-ului etichete distincte in vederea evitarii definitiilor multiple.

2.2.2.9. Directivele .t414 .t800

Sintaxa :

```

.t414
.t800

```

Au rolul de a specifica transputerul pentru care se face asamblarea codului.

Initial se considera implicit T414.

2.2.3. Nume macro-uri

Aparitia in cimpul operator a unui nume care nu e mnemonica sau nume directiva, determina cautarea numelui respectiv in lista de macro-definitii. Apelul unui macro are ca efect inserarea liniilor corpului macro cu substituirea simpla a parametrilor formali cu cei actuali. De asemenea, se genereaza etichetele locale daca exista o directiva .label.

2.3. Cimpul operand

Contine o expresie aritmetica formata din constante, nume simbolice si operatori. Se folosesc urmasorii operatori in ordinea precedentei :

()	- paranteze stinga, dreapta
~	- complement fata de 2 si minus unar
>> <<	- deplasare dreapta, stinga
&	- si sau (pe bit)
* /	- produs, impartire
+ -	- adunare, scadere

Ca operandi se pot folosi nume simbolice definite cu directiva .eq sau ca etichete si constante numerice.

Constantele se specifica astfel :

- constante zecimale	dddd	Ex: 1234
- constante hexazecimale	shhhh	Ex: \$12A6

- constante octale	80000	Ex: 3177
- constante binare	1b000	Ex: 11001
- constante ASCII	"aaa"	Ex: "sir"

2.4. Cimpul "comentarii"

Sintaxa : !"comentariu"

Este optional si e ignorat de asamblor . Se introduce prin caracterul "!" in prima coloana.

3. STRUCTURA DE PRINCIPIU A ASAMBLORULUI OPTIMIZAREA CODULUI

Asamblorul lucreaza in 4 faze principale: preprocesare, trecerea 1, optimizare, trecerea 2.

Preprocesarea este destinata tratarii directivelor de asamblare conditionata si macroprocesarii. Trecerea 1, in principal, asigura crearea tabelii de simbolii, iar in trecerea a doua se genereaza codul.

Necesitatea unei faze de optimizare este ilustrata de urmatoarele exemple:

Exemplul 1:

Instructiunea ldc \$A12 se poate traduce in oricare din urmatoarele variante echivalente:

2A	prefix \$A	20	prefix \$0
21	prefix \$1	20	prefix \$0
42	prefix \$2	20	prefix \$0
		20	prefix \$0
		20	prefix \$0
		2A	prefix \$A
		21	prefix \$1
		42	ldc \$2

Prima varianta nu incarca toti cei 32 de biti ai registrului operand. A doua varianta, care completeaza cu 0 la stanga registrul operand, are avantajul ca indiferent de operand, lungimea codului este constanta, dar nu este optima.

Dependenta lungimii codului generat de valoarea operandului determina o serie de dificultati.

Exemplul 2:

```
1      et1:
2          cj    et2
3      et3: j    et1
      ...
n      et2:
```

Secventa anterioara e dificil de tradus din urmatoarele motive: la prima trecere, in tratarea liniei 2, valoarea simbolului (adresei) "et2" nu este cunoscuta si nu se poate

determina lungimea codului generat pentru instrucțiunea "et2", decît dacă se considera offsetul pe 32 de biți.

Contorul de locații nu se poate incrementa, deci simbolul "et3" nu se poate introduce în tabela de simboluri cu valoarea corespunzătoare. De asemenea, nu se poate calcula offsetul et3-et1 necesar în generarea codului pentru instrucțiunea "j et1".

O soluție imediată pentru completarea tabelului de simboluri cu valori contor-program (p_c) corecte este sugerată în (11) și constă în mai multe treceri asupra programului. La prima trecere se realizează doar o estimare a valorilor p_c ale simbolurilor din tabel. Următoarele treceri nu generează cod, ci recalculează aceste valori obținind estimări mai bune pînă la obținerea valorilor p_c corecte. Soluția este inacceptabilă sub raportul timpului de asamblare.

În asamblorul descris de prezenta lucrare am implementat o metodă mai elegantă care evită trecerile multiple și, în plus, permite expresii aritmetice oarecare în timpul operand.

Principal, metoda constă în următoarele:

1. La evaluarea unei expresii aritmetice care apare ca operand se recalculează și un atribut expresie care precizează dacă valoarea ei este absolută (formată numai din constante) sau depinde de una sau mai multe valori de tip adresă (etichetă).

2. Pentru expresiile care nu sînt absolute se introduc în tabela de simboluri pseudoetichete, iar o formă a expresiei se păstrează în vederea recalcularilor ulterioare.

3. La prima trecere se considera că valorile tuturor expresiilor care nu pot fi calculate sînt reprezentabile pe 4 biți. Se obține astfel o primă estimare a valorilor din tabela de simboluri.

4. Urmează o fază de optimizare în care se parcurge tabela de simboluri de mai multe ori pînă la stabilizarea valorilor p_c.

În acest scop tabela de simboluri este organizată ca o structură dublu înlantuită. Tabela poate fi exploatată atît ca o structură hash (la interogare), cit și ca o listă înlantuită simplă de simboluri a căror ordine în listă este ordinea apariției lor în program. Pe parcursul unei treceri prin tabela de simboluri (listă) se identifică prin intermediul pseudoetichetelor prima expresie care nu este absolută și se reevaluează. Diferența pozitivă sau negativă față de valoarea anterioară se adună la toți simbolii de tip etichetă care urmează în program.

Procesul continuă cu următoarea expresie care nu este absolută, pînă la parcurgerea completă a tabelului de simboluri. Se obține astfel o estimare mai bună.

Se reia cu o nouă parcurgere pînă nu mai apar diferențe.

Ultimile asignări ale valorilor p_c la simboluri sînt folosite în trecerea a doua care generează codul.

Concluzii

Lucrarea are un caracter informativ si nu prezinta toate detaliile de proiectare si implementare. Acestea pot fi gasite in [2] sau pot fi cerute autorilor. Algoritmul de optimizare este original si are caracter de noutate.

Eventualele neclaritati sau omisiuni nu sunt voluntare si se datoreaza spatiului limitat de prezentare.

Bibliografie

- [1] INMOS-Transputer instruction set - a compiler writer's guide
- [2] I.I.T. Fil. Academiei, Iasi - MAT Macroasamblor pentru transputer T414, T800 - Specificatii de definitie

SISTEME EXPERT IN COMUNICATIE CU BAZE DE DATE

Autori: dr. ing. D. Carstalea
Institutul Politehnic Bucuresti

1. INTRODUCERE

Lucrarea prezinta o modalitate prin care pot fi crescute capacitatile unui sistem de baze de date (DBMS) prin utilizarea unui sistem expert (SE). Sint cunoscute in principal trei moduri de integrare a sistemelor expert si bazelor de date obtinand ceea ce se numeste baze de date inteligente, sisteme expert cu capacitate crescuta si sisteme expert/baze de date comunicative. Principala diferenta intre cele trei modele este marcata de modul de cuplare si de alocare a controlului in sistemele ce le integreaza. Principala problema a bazelor de date inteligente este cea a accesului la inregistrari dupa diferite criterii, alegerea celui mai bun criteriu de acces putind fi formulata ca o problema de decizie a celei mai bune selectii dintr-un subset al alternativelor posibile. Tehnicile de utilitate teoretica cum sint conflictuale de rezolutie, metodele interactive si cele cu intindere mare pot fi inlocuite cu succes in multe din problemele complicate specifice bazelor de date. Conceptele 'fuzzy' utilizate inevitabil atit in sistemele expert cit si in rationarea umana pentru luarea deciziilor se pot utiliza cu succes in problematica bazelor de date.

Un sistem de baze de date impreuna cu un sistem expert bazat pe logica 'fuzzy' poate manipula concepte vagi cerute de ambele aplicatii pe linga functiile normale alocate acestora. Aplicatiile specifice acestor categorii de probleme se vor construi intr-un mediu constituit dintr-un, sistem expert bazat pe logica 'fuzzy' impreuna cu un modul de interfatare la un program specific de baze de date.

2. INTERACTIUNEA SISTEM EXPERT BAZE DE DATE

O baza de date este o colectie de date intr-o forma organizata ce poate furniza date catre sistemul expert prin intermediul sistemului de manipulare a datelor. Exista mai multe posibilitati de interactiune intre sistemul expert si baza de date, interactiuni ce pot fi grupate in trei categorii in conjunctie cu gradul de cuplare si alocare al controlului. Se disting astfel sistemele expert cu capacitate marita, bazele de date inteligente si sisteme expert/baze de date in comunicatie. Desigur ca aceste trei categorii pot fi impartite in viitor in alte clase si nivele asa cum se arata in fig.1.

In sistemele expert cu capacitate marita facilitatile extinse de manipulare a datelor sint implementate la nivelul sistemului expert. In acest caz datele sint furnizate sistemului expert de baza de date atunci cind sint cerute, fie sint transmise sistemului expert inaintea oricarei activitati a acestuia ca la transferul imaginii intr-o fotografie la minut. Este modul in

care baza de date este pusa la dispozitia sistemului expert in totalitate.

Prin inglobarea componentelor deductive intr-un sistem de baze de date se obtin bazele de date deductive si inteligente. Componenta deductiva poate creste eficienta si functionalitatea sistemului de baze de date prin optimizarea cerintelor, pastrarea integritatii si a altor facilitati. Formal o baza de date deductiva este definita ca un triplet

$D = \langle Th, IC, DO \rangle$ in care,

* Th reprezinta o teorie cu un set finit de axiome logice, si un set finit de metareguli ce descriu modul de operare cu axiomele teoriei particulare.

* IC reprezinta un set finit de restrictii de integritate ce trebuiesc sa fie satisfacute de baza de date, constind dintr-un set finit de formule ale calculului cu predicate.

* DO reprezinta setul finit al constantelor peste care teoria este definita.

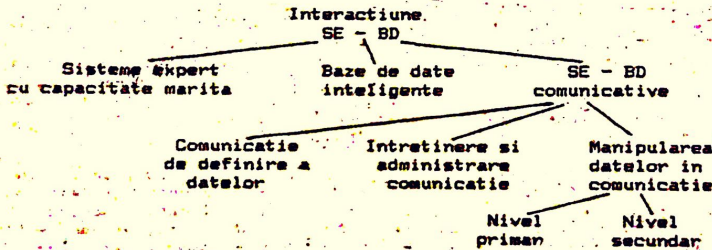


Fig. 1.

Reuniunea $\langle Th \cup IC \rangle$ dintre teoria $\langle Th \rangle$ si restrictiile de integritate $\langle IC \rangle$ trebuie sa satisfaca conditia de consistenta. Formulele atomice si regulile de deductie sint doua dintre axiomele tipice ale unei teorii. Formulele atomice intre n-upluri si relatii formeaza baza de date extensionala pe cind regulile deductive, reprezentate prin declaratii in forma logica determina o baza de date intensionala.

Ca exemplu se considera o baza de date formata din doua relatii extensionale: $aria\_locatie(id\_arie, judet)$ si $trafic\_conditii(judet, ruta)$, pentru care se poate defini o relatie intensionala ce arata conditiile de trafic pentru aria in discutie astfel:

```

V id_aria V judet V ruta
aria_locatie(id_aria, judet) & (trafic_conditii(judet, ruta)
----> aria_trafic_conditii(id_aria, ruta)
  
```

A treia categorie sisteme expert/baze de date comunicative are

avantajul asigurării independentei celor două componente, fiind posibile câteva forme de comunicare între acestea. Utilizarea unei baze de date existente face ca gradul de complexitate al comunicății între SE și DB să fie mult mai mare. La rândul său comunicăția poate fi clasificată în trei clase:

- * comunicăția de definire a datelor, prin care SE poate altera structuri și componente ale BD. Dacă cunoștințele privind dezvoltarea BD cum sint cele despre normalizare sint continue de sistemul expert atunci o BD poate fi construită sau modificată automat.

- * întreținerea bazei de date și administrarea comunicăției este situația în care SE poate rezolva sarcini administrative cum sint analiza stocării, gestionare backup, reorganizarea și întreținerea datelor.

- * manipularea datelor în comunicăție, prin care SE trimite comenzi pentru adăugare, inserare, ștergere către baza de date. Utilizarea numai a acestui tip de schimb de informație face ca un SE să aibă forma unei baze de date deductive. O instrucțiune logică a unei relații intensionale într-o bază de date deductivă este de fapt o regulă ce poate fi împărțită în două părți: prima parte este o definiție a unui n-uplu scris în termenii calculului relațional ce este echivalent cu partea IF a regulii, a doua parte fiind data de numele și atributele n-uplului pentru relația intensională ce este echivalentă cu partea THEN. Este motivul pentru care instrucțiunile logice pentru relațiile intensionale într-o bază de date deductivă pot fi înlocuite prin reguli într-un SE, împreună cu un limbaj de cereri relaționale suportat de DBMS.

Pentru a treia clasă de comunicare sint necesare două nivele ale manipularii datelor. Primul nivel este specific bazelor de date, fără să țină cont de operațiile în SE. Cu acest nivel SE manipulează și selectează înregistrări prin el însuși făcându-l echivalent cu un SE de mare capacitate. Al doilea nivel presupune utilizarea operatorilor de înalt nivel și a altor limbaje suportate de DBMS. La acest nivel pentru accesul la date SE transmite cereri către DBMS utilizând partea IF a regulii ca o condiție de selecție. În acest mod SE nu va selecta sau sorta date prin el însuși.

Lucrarea de față este destinată celei de a treia clase de cuplare care asigură avantajul utilizării unei BD existente, deoarece nu este practic să se construiască un nou sistem de BD în care să se transfere datele înainte de a fi utilizat. Astfel DBMS împreună cu setul de date este văzut ca un sistem independent. Un modul special care asigură interfata între SE și DBMS este necesar, modul care acceptă cereri fuzzy de la SE sau poate fi utilizat independent dacă utilizatorul furnizează cereri în limbajul acceptat.

3. ARHITECTURA SISTEMULUI

Un SE 'fuzzy' pentru aplicații cu BD (SEBD) este constituit în principal din trei componente și anume:

- * Sistemul Expert Fuzzy (SEF)

- * Modulul de Comunicare cu Baza de Date (MCBD)
- * Baza de Date (BD)

Structura SEBD este prezentată în fig.2. SEF are o structură tipică SE bazat pe reguli. Este compus dintr-un modul de achiziția cunoștințelor, un mecanism de inferență și o bază de cunoștințe fuzzy. Cunoștințele în domeniu sunt extrase de la experții umani și transferate sistemului prin intermediul modulului de achiziția a cunoștințelor. Regulile și faptele pot fi stocate în baza de cunoștințe prin utilizarea și a termenilor fuzzy. Mecanismul de inferență este responsabil de utilizarea cunoștințelor și deducerea concluziilor bazate pe cunoașterea exactă sau inexactă.

MCBD asigură interfata între SEF și BD și face ca SEF să poată utiliza BD. Sistemul poate manipula date fuzzy ce pot fi direct stocate în baza de date și sunt utilizate la raționarea fuzzy în timpul consultării. Faptul că SEF manipulează cereri lingvistice fuzzy este foarte important în luarea deciziilor pentru multe probleme reale.

4. REPREZENTAREA CUNOȘTINTELOR ȘI INFERENȚE

În SEF așa cum s-a prezentat mai sus cunoștințele sunt reprezentate prin reguli de formă:

IF propoziție-antecedent
THEN propoziție-succedent
WITH CERTAINTY factor-certitudine

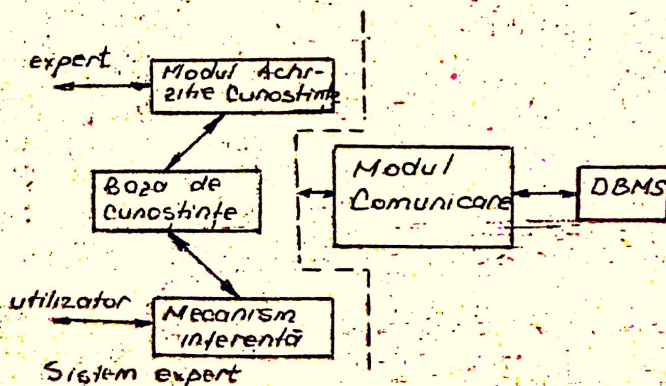


Fig.2

Astfel fiecare propoziție este reprezentată ca "obiect" cu

"valoare", ca de exemplu:

```
IF      hirtia indicatoare este rosie
THEN   solutia este acida
WITH CERTAINTY 1.0
```

Astfel un obiect ce este o unitate de baza in baza de cunostinte (BC) poseda un numar de proprietati ce sint transmise in faza de achizitie a cunostintelor. Se dau mai jos citeva exemple ale acestor proprietati:

Proprietate	Semnificatie
Tip	Tipul obiectului
Flag-cerere	Valoarea este obtinuta prin intrebari adresate utilizatorului sau prin deducerea din alte reguli si fapte
Flag-fuzzy	Indica faptul ca obiectul are o valoare fuzzy sau nu

Asa cum in rationarea umana se utilizeaza concepte fuzzy, SEF permite termeni fuzzy in reguli si este capabil sa realizeze rationamente aproximative, ca de exemplu:

```
Regula  IF      masina este puternica
        THEN   dezvolta viteza mare
Fapte   Masina este foarte puternica
```

Concluzie Viteza dezvoltata este foarte mare

In acest exemplu puternic, foarte puternic si viteza sint termeni fuzzy. Fiecarui termen fuzzy ii corespunde un set fuzzy, setul fuzzy reprezentind concluziile este obtinut prin operatiile realizate intre setul fuzzy reprezentind faptele si relatiile fuzzy reprezentate in regula. Propozitii multiple si termeni mixti fuzzy si non-fuzzy sint permisi in partea antecedent si succedent a regulii.

O metoda de rationare este cea de inlantuire inainte, asa ca atunci cind valoarea unui obiect este ceruta SEF incearca mai intii sa o caute in baza de date. Daca valoarea este lipsa atunci sistemul invita utilizatorul sa o introduca direct determinind astfel o legatura dinamica intre componentele sistemului. Datele sint cautate in baza de date numai atunci cind este necesar un proces de inferenta la nivelul SE. Ca exemplu se poate considera situatia starii sanatatii si a sansei de insanatosire a unui bolnav. Daca starea de sanatate este apreciata dupa greutate corelata cu inaltime se poate aprecia sansa insanatosirii ca fiind foarte mica, mica, mare, foarte mare. Desigur ca este important de observat ca termenii fuzzy sint recomandabili a fi utilizati cind este necesara utilizarea unor descriptori numerici cu mai multe valori.

O caracteristica a acestui sistem este abilitatea de acces la informatiile relevante din baza de date la sfirsitul consultarii

acesteia. Aceasta este obtinuta prin generarea unui set de cereri fuzzy catre baza de date dupa ce sistemul expert trage concluzia. In timpul procesului de inferenta, restrictiile ce constituie cereri la baza de date sint culese. Astfel ca, in regula

```
IF                                persoana este adult
THEN BET_RESTR                  virsta)21 GREUTATEA 0.8 AND
                                poate vota
```

SET_RESTR este o actiune ce indica faptul ca urmatoarea propozitie este o restrictie. Daca regula este evaluata cu succes fraza virsta)21 este extrasa si devine una din restrictii. GREUTATEA este un cuvint cheie ce specifica faptul ca factorul greutate este 0.8 in acest caz, el fiind in general reprezentat ca un set fuzzy cu valori reale cuprinse intre 0 si 1. Din inaintuarea tuturor regulilor se obtine cererea pentru baza de date. Cererile fuzzy returneaza o gama larga de informatii a caror apartenenta este determinata de continutul bazei de date. Pentru formularea:

```
(sportiv_rec(numa) () (and(grad"bun" 1.0)))
```

adica lista numelor sportivilor cu rezultate bune, pentru care a fost definita distributia fuzzy dupa punctaj astfel

grad()	0	10	20	30	40	50	60	70	80	90	100
set fuzzy	0	0	0	0	0	0.1	0.2	0.4	0.7	0.9	1.0

se obtine o lista a sportivilor ordonata dupa valoarea variabilei fuzzy asociate. Se observa ca valorile gradelor de participare sint determinate de distributia participarii datelor stocate in baza de date.

Daca sint aplicabile mai mult de o conditie fuzzy apartenenta la o clasa este obtinuta prin combinarea conditiilor rezultate in mai multe moduri:

1. Calculul minimum/maximum (and/or)
2. Combinarea evidentei ca in teoria confirmarii (comb)
3. Interogare si capturare marime (poll)

Daca se presupune ca $a_1, a_2, \dots, a_n$ sint gradele de apartenenta, se obtine dupa aplicarea operatorilor definiti mai sus.

AND $f(a_1, a_2, \dots, a_n) = \min(a_1, a_2, \dots, a_n)$

OR $f(a_1, a_2, \dots, a_n) = \max(a_1, a_2, \dots, a_n)$

COMB $f(a_1, a_2) = a_1 + a_2(1 - a_1)$

$$f(a_1, a_2, \dots, a_n) = a_1 + a_2(1 - a_1) + a_3(1 - a_1 - a_2) + \dots + a_n(1 - a_1 - a_2 - \dots - a_{n-1})$$

POLL $f(a_1, a_2, \dots, a_n) = (a_1, a_2, \dots, a_n) / n$

Orice combinație a celor patru operatori este permisă a fi folosită într-o expresie fuzzy. Inregistrările cu gradul de apartenență final zero sunt eliminate din setul de ieșire. Logica fuzzy este superioară metodelor de selecție multicriteriale prin flexibilitatea manipularii relațiilor dintre aceste criterii. De asemenea, criteriile fuzzy sunt subiective și trebuie formulate cu multă grijă.

Modulul de comunicare cu baza de date preia cererile formulate de SEF pentru obținerea informațiilor din baza de date, activitățile sale putând fi etapizate în trei faze:

- * traducerea cererilor SEF în limbajul specific BD.
- * preluarea datelor de la BD.
- * procesarea datelor cerute pentru încadrarea în clase de apartenență, ordonarea acestora în ordine descrescătoare a gradului de apartenență și eliminarea celor cu grad de apartenență zero.

Sintaxa cererilor fuzzy utilizate de SE este următoarea:

```

QUERY          ::= ((nume_relatie) LIST_SIMB, COND_FUZZY,
                      NON_FUZZY_CON)
LIST_SIMB      ::= ST_EXP I ((num_cimp1), ..., (num_cimp1))
COND_FUZZY     ::= FUZZY_CON I nil
ST_EXP        ::= ((ST_OP NON_FUZZY_CON)
NON_FUZZY_CON ::= ((OPERATOR TERM1....TERMJ)
FUZZY_CON      ::= ((FUZZY_OP FUZZY_EX1....FUZZY_EX2)
ST_OP          ::= max(num_cimp) I
                  min(num_cimp) I
                  total(num_cimp) I
                  average(num_cimp) I
                  count
TERM           ::= NON_FUZZY_CON
                  (COMPARATOR (num_cimp) (valoare)) I
                  (COMPARATOR (num_cimp) ST_EXP)
FUZZY_EX       ::= FUZZY_CON I FUZZY_TERM
FUZZY_TERM     ::= ((num_cimp) (membru al distributiei)
OPERATOR       ::= and I or
FUZZY_OP       ::= and I or I comb I poll
COMPARATOR     ::= ((, (=, >, <))=)

```

Sînt necesare cîteva precizări asupra caracteristicilor DBMS pentru a se asigura integrarea în acest sistem. S-a presupus că DBMS nu este specializat dar accesul la informații se realizează prin combinarea comenzilor START_STREAM, FETCH, GET, sau echivalente. Inregistrările ce satisfac criteriul de selecție sînt incluse în sirul inregistrărilor specificate cînd se execută START_STREAM, un pointer de sir este plasat înaintea primei inregistrări a sirului. Avansul în sir se realizează prin comanda FETCH, iar valoarea și numele cîmpului poate fi obținut prin GET. Procesul se repetă pînă cînd toate inregistrările din sir sînt explorate.

5. CONCLUZII

În acest mod datele pentru inferență sînt obținute de la o bază de date într-o formă organizată permițînd prelucrarea cu sistemul expert a unei mari cantități de date. Acest mod de abordare, cît și utilizarea raționamentelor fuzzy poate oferi suportul de rezolvare a multor probleme ale lumii reale. O aplicație ce devine interesantă este cea a diagnosticului genetic unde se dispune de o bază de date organizată dar care ocupă 14 Mo, cantitate de informație ce nu poate fi utilizată în alt mod.

6. BIBLIOGRAFIE

1. Teză de doctorat, D. Carstoiu 1989.
2. Rule-based Expert Systems, B. G. Buchanan, 1984.
3. Fuzzy Set Theory and its Applications, H. J. Zimmermann, 1986.
4. Expert Systems and Fuzzy Systems, C. Negoita, 1985.
5. Data & Knowledge Engineering, 1989-1990, North-Holland.

SISTEME SUPORT DE DECIZIE CU APLICATII IN MEDICINA

Autori: Dr. ing. Dorin Carsteiu
Institutul Politehnic Bucuresti

1. INTRODUCERE

Inteligența artificială este un domeniu relativ nou al științei calculatoarelor ce își propune să rezolve problemele într-o manieră apropiată de modul de gândire uman. Dintre instrumentele de lucru ale inteligenței artificiale, sistemele expert ocupă un loc central mai ales prin varietatea aplicațiilor lor. Dezvoltate în ultimul deceniu, sistemele expert au revoluționat viziunea referitoare la tipurile de probleme ce pot fi rezolvate folosind calculatorul.

Organizate puțin diferit față de familiarele programe algoritmice, sistemele expert pot folosi tehnici euristice și raționamente "fuzzy" pentru a ajunge la rezultate. În plus astfel de sisteme încorporează un volum mare de cunoștințe asociate unor domenii specifice. Sistemele expert sunt programe foarte performante în domeniile lor înguste de expertiză pentru care au fost elaborate. Bine întreținute, relativ ieftine, cu mare grad de portabilitate și fiabilitate ridicată, devin din ce în ce mai eficiente cu trecerea timpului.

2. SISTEME SUPORT DE DECIZIE

O categorie mai largă privind utilizări frecvente ale tehnicii de calcul în domeniul medical este cea a sistemelor suport de decizie (SSD). În ciuda entuziasmului manifestat de mulți utilizatori, în ultimii ani dezvoltările de succes se referă la domenii restrinse și bine definite.

Ca regulă generală, decizia umană este o operație complexă ce presupune întotdeauna o înțelegere profundă a procesului. Deliberările sunt adesea neclare și părți ale fenomenului nu pot fi conchise individual. Construcția SSD poate de cele mai multe ori să ceară calculatoarelor facilități de simulare a comportamentului uman. În viitor, informațiile cerute de calculatoare cer compresia colecțiilor de date prin tehnici structurale ce sunt incomode din punctul de vedere al utilizatorului. Astfel, dezvoltarea unui SSD valid necesită un proces lung cu reveniri și noi acumulări de date. Un exemplu graitor este proiectul INTERNIST ce a fost dezvoltat continuu mai mulți ani. Ca și la alte produse, calitatea unui astfel de sistem trebuie să fie analizată și validată în urma exploatării într-un timp îndelungat. În cazul sistemelor cu aplicații clinice validarea are un important caracter particular. Datorită faptului că sistemele suport intenționează să fie un ultim pas în decizia medicală, multe informații din diverse surse vor fi combinate în deciziile clinice de diagnosticare și terapie. Validarea unui astfel de task devine foarte dificilă datorită complexității

deosebite a sistemului. Numai un sistem ce funcționează într-un context particularizat pentru un grup individualizat de utilizatori poate beneficia de opinii calificate asupra capacității sale.

Sigur că pentru o analiză atentă a SSD cu aplicații clinice trebuie examinate după destinație: în diagnosticare, tratament, prognoza, etc.. De asemenea răspunsul la o serie de întrebări trebuie să țină seama de toți factorii implicați în utilizarea sistemului, adică, pacient, informatician, medic, societate. Considerentele legate de utilizare presupun evaluarea metodelor fundamentale referitoare la algoritmi utilizați, cunoștințele stocate și sursa acestora, cit și a caracteristicilor operaționale. Există la ora actuală programe de dezvoltare a mediilor sistem pentru limbaje orientate profesional, programe ce includ cercetări și colaborări multidisciplinare în sfera informaticii și medicinei.

În context clinic, obiectivele utilizării SSD trebuie judecate independent de performanțele sistemului, realizând pe cât posibil compromisul între complexitatea problemelor medicale și simplitatea manipulării noilor mecanisme de decizie. Multe dintre sistemele utilizate în domeniul medical sunt destinate furnizării cunoștințelor specializate, pornind de la premisa că fie utilizatorul este înzestrat cu o cantitate mare de informații, fie că problema duce la situații nefamiliare acestuia. Din acest motiv se pot distinge patru tipuri de sisteme:

- * sisteme vizând condensarea informației pentru facilitarea prelucrărilor ulterioare, având avantajul lărgirii capacității personalului uman în rezolvarea de probleme complexe.

- * sisteme capabile să ofere utilizatorilor nespecializați suportul deciziilor, purtând din acest motiv și numele de sisteme consultative.

- * sisteme de învățare utilizate pentru educarea și perfecționarea utilizatorilor, cu avantajul creșterii competenței profesionale a acestora.

- * sisteme de informare și reorganizare, cu avantajul accesului simplu la literatura specifică problemei.

Scopul final al aplicării oricărei noi tehnologii este creșterea grijii față de sănătate, în sensul luării de către medic a celei mai bune sau a celei mai puțin eronate decizii. Desigur că, utilizarea unui astfel de sistem va determina doctorul să cheltuiască mai mult timp cu pacientul, influențând și modificări în structura serviciilor de sănătate, noi analize ale raportului cost/eficiență.

Cum esența unor astfel de sisteme este tocmai stocarea cunoștințelor, se remarcă validitatea mult mai mare a cunoștințelor față de date. Este adevărat că aceste cunoștințe au diverse costuri. Se poate defini o cunoștință ca o abstractizare a experienței obținute de om, ce este direct sau indirect derivată din date și informații. Domeniile cunoașterii se pot împărți în două și anume domenii închise și domenii deschise. Un domeniu închis este rar și foarte restrâns și nu poate acoperi, nici

parțial, un domeniu clinic. Este deci necesar ca utilizatorii să cunoască caracteristicile propriului domeniu, focalizarea și identificarea cerințelor aplicației sistemice, tehnicile probabilistice și multimile vagi.

3. EXTRAGEREA CUNOSTINTELOR

Cunostintele specifice domeniului de expertiza sunt stocate în diverse moduri, modelele cel mai des întâlnite fiind regulile de producție, rețelele semantice precum și reprezentare orientată obiect. Deducția în forma sa cea mai simplă constă din două părți: partea din stînga, premiza sau ipoteza și partea din dreapta, consecința sau acțiunea. Astfel, ipoteza implică concluzia sau premiza acțiunea. Diferența între aceste două categorii este minimă în sensul că satisfacerea premisei declanșează automat acțiunea.

De exemplu, un medic știe că tensiunea normală pentru un bărbat de 30 ani cu înălțimea de 1,80 m care cântărește 75 kg este de 11/7. Acest fapt se poate exprima ca o regulă prin care se atribuie o valoare unei variabile TENSORM în funcție de valoarea atribuită variabilei VIRSTA.

```
IF:   VIRSTA = 30
      AND INALTIMEA = "1,80"
      AND GREUTATEA = 75
THEN: TENSORM = "11/7"
```

Procesorul de inferență evaluează premiza determinând valoarea de adevăr a expresiei, și în funcție de aceasta, declanșează acțiunea. Un sistem expert simplu folosit pentru determinarea greutății și tensiunii normale a unei persoane pentru care se dau vîrsta, sexul și înălțimea conține relativ puține reguli în timp ce un sistem expert pentru aplicații largi poate conține sute de reguli.

Cunostintele de diagnosticare sînt esențialmente empirice și deriva din observațiile asupra diferiților pacienți individuali. Deci prima evaluare a surselor de cunoaștere este marcată de observarea bazei de cunostințe. Este necesar în acest moment să se stabilească criteriile de incluziune și excluziune a elementelor de cunoaștere, știind că baza de cunostințe este cea care afectează puternic validitatea SSD.

Datele despre pacienți sînt colectate prospectiv și retrospectiv. De regulă înregistrările retrospective au o altă destinație și sînt incomplete. Rezultatele negative sînt rareori înregistrate.

În cartile medicale, articole, alte publicații și lucrări științifice se reprezintă multiple cunostințe despre modul de stabilire a diagnosticului. Trebuie remarcat faptul că informațiile provenind din carti au ca punct de plecare simptomatologia fără a oferi explicații pe traseul invers: diagnostic - simptom. Experiența specialiștilor individuali sau de grup reflectă raționamentul pentru diagnosticarea în domeniu.

specificat, operind dupe anumite scenarii, de multe ori diferite dar cu aceleasi rezultate. Datele sint usor organizate prin program, informatiile obtinute in acest mod putind suplimenta cunostintele achizitionate. Un aspect important se refera la tratarea cazurilor inchise si a celor extreme. Sint relevante aspectele legate de clasificarea pacientilor dupa virsta, sex, etnii, criterii geografice si socioeconomice.

Cele mai multe SSD au fost construite pe baza datelor colectate de un grup mic de cercetatori. Transferul de la un grup de utilizatori la altul, prin caracterul diferit al observatiilor poate duce la cresterea acuratetii rezultatelor. Reproducibilitatea observatiilor va trebui evaluata ca parte a acizitiei de cunostinte si apreciata astfel pentru adaugarea de noi date pe timpul testarii de sistem. Gradul transferabilitatii bazei de cunostinte depinde de mai multi factori, valorile predictive ale SSD trebuind sa fie testate cind sint aplicate unui nou mediu.

Pentru validarea rezultatelor SSD este necesar sa se cunoasca citiva rezultate standard, general acceptate, cu care sa se compare iesirile sistem. Sint posibile doua situatii de acceptare atunci cind se obtine un raspuns corect pentru problema cu obiectivul cautat sau cind concluziile corespund in mare parte cu cele ale expertului uman (sau grupurilor de experti umani). Este de asemenea dificil de clasificat rezultatele SSD ca fiind corecte sau false, intrucit solutia unei probleme se obtine ca o combinatie de rationamente si propozitii cu diferite grade de certitudine. Concluzii alternative pot fi corecte la diferite nivele de incredere si o solutie corecta la un caz poate acompania un diagnostic incorect.

Este greu de statuat faptul ca opiniile personale sau practica colectiva serveste ca standard de baza intrucit acestea pot varia de la loc la loc prin factori relativi la traditie, considerente politice si socioeconomice, nu neaparat legate de stiinta medicala.

4. EVALUARI ALE MECANISMULUI DE INFERENTA

Sistemele expert cadru permit construirea seturilor de reguli de diverse complexitati, complexitatea fiind in general limitata de structura hardware. Nu toate cunostintele din domeniul de expertiza sint luate in considerare cind o anumita premisa este indeplinita. Una dintre principalele caracteristici a sistemelor expert este cea a determinarii acelor cunostinte ce sint relevante pentru situatia data. In limbajele de programare traditionale nu se pune problema determinarii instructiunii urmatoare de executat deoarece toate instructiunile sint executate consecutiv. O structura conditionala care foloseste instructiuni IF-THEN/ELSE, GOTO, bucle, este un caz extins de natura secventiala sau procedurala a limbajului de programare.

Procesorul de inferenta alege din informatiile furnizate de utilizator pe acelea ce sint aplicabile in rezolvarea problemei.

Exista doua strategii de control ce se pot folosi in determinarea regulii de selectie, inlantuirea inainte si inlantuirea inapoi. Pentru ilustrare se va considera un sistem expert bazat pe reguli de productie.

Inlantuirea inapoi este metoda de control cea mai utilizata in sistemele expert. In acest caz sistemul incepe cu scopul propus, apoi examineaza setul de reguli pentru a gasi prima regula care poate sa atinga scopul propus. Daca premiza primei reguli este TRUE, atunci sistemul realizeaza actiunea corespunzatoare. Daca premiza este FALSE, atunci procesorul de inferenta considera regula urmatoare pentru atingerea scopului. Daca premiza acesteia este TRUE atunci regula este executata. Utilizatorul poate alege inlantuirea inapoi in cazul in care procesorul de inferenta al sistemului cauta un scop. Gasirea valorii devine un scop nou (temporar) al programului. Prin inlantuire inapoi, pentru a determina informatia necesara, se cauta in lista de reguli o regula care atribuie o valoare pentru noul scop - de aici "inlantuire inapoi" - pana ce setul de reguli este epuizat. Se analizeaza premiza noii reguli conform necesitatilor curente. Daca valoarea de adevar a premizei nu poate fi stabilita, fiind necunoscuta, devine un nou scop si ciclul se repeta. Un sistem expert permite reluarea pas cu pas a procesului prin care procesorul de inferenta a ajuns la decizia sau solutia particulara. Directia de inlantuire, inainte sau inapoi, este cea mai importanta contributie la capacitatea sistemului expert de a raspunde intrebarilor puse.

Inlantuirea inainte porneste de la premiza la actiune pana cind variabilei scop i se atribuie o valoare. Procesorul de inferenta incepe prin a cauta in setul de reguli prima regula a carei premiza este TRUE. Aceasta va fi executata si intoarce un rezultat ce va putea fi folosit pentru testarea altor reguli. Aceasta valoare devine o parte din baza de date si este disponibila si pentru alte reguli. La urmatoarea trecere prin setul de reguli, se cauta prima regula a carei premiza este TRUE.

Prin mecanismul de inferenta se arata cum utilizeaza sistemul cunostintele si datele clinice pentru a construi rezultatul. Un mecanism de inferenta poate fi impartit in trei mari categorii dupa modul in care aceasta se realizeaza:

- * inferenta bazata pe similaritate, incluzind date statistice.
- * inferenta cauzala.
- * inferenta deductiva bazata pe reguli explicite.

Mecanismul de inferenta bazat pe similaritate are mai ales o importanta istorica, primele sisteme pentru diagnoza medicala au fost pur empirice si statistice. La acestea frecventa simptomelor si semnalmentelor au fost tabelate alaturi de conditiile de adevar obtinute din observarea pacientilor. Aceasta abordare a fost motivata prin presupunerea ca daca s-a obtinut odata un succes se poate asuma riscul reproductibilitatii. In Jurul anilor

1975 s-a dovedit ca aceasta abordare produce bune rezultate in domeniul clinice inguste. Din acest motiv cele mai multe cercetari imbraca in sistemele expert reale combinatii intre factori statistici, cunostinte teoretice si opinia expertului. Toate metodele statistice referitoare la noii pacienti pornesc de la presupunerea similaritatii. Metodele statistice difera dupa maniera in care se defineste similitudinea si deci, de modul in care se exploreaza datele din baza de date. Pot fi facute diferite presupuneri de distributie. Procedurile statistice acopera intre 5 si 50 % din cazurile similare si permit reducerea bazei de date pornind de la principiul similitudinii informatiilor. Aceste proceduri pot fi impartite in doua categorii, ambele pornind de la probabilitatea condicionala a profilului de sindromuri asociate pacientului. O clasa de modele pornesc de la probabilitatea directa, pe cind cealalta pastreaza descrierea statistica a simptomatologiei bolii, utilizand probabilitatea inversa calculata dupa formula inversa Bayes.

Mecanismele de inferenta cauzale utilizeaza declaratii cauzale calitative, (insulina KI determina modificari intracelulare) sau modele fiziologice si farmacocinetice. Cunostintele de acest tip sint reprezentate in baza de cunostinte si se numesc cunostinte 'profunde'. Sistemul consta dintr-o retea complexa de interconectare a regulilor de acest tip, regulile simple putind interactiona in diverse moduri, asa ca, prin inspectarea bazei de cunostinte fiecare regula poate sa aduca o contributie majora la rezultat. Retelele cauzale pot contine bucle prin faptul ca insasi mecanismul cauzal pe care il reprezinta implica mai multe reevaluari. Pot fi reprezentati astfel doi factori anatomici: adiacenta si conexiunile functionale.

Un exemplu in acest sens este NEUROLOGIST, sistem pentru localizarea placilor in scleroza multipla, in care o placa poate afecta traiectele auxiliare. Conexiunile anatomice determina legaturi false cum este cazul invaziilor cancerigene. Cu toate ca acestea utilizeaza cunostinte anatomice, o anatomie pe calculator nu a fost realizata inca.

5. ANALIZA CONSECINTELOR

In ultimul timp multe SSD au fost introduse in practica clinica obisnuita. Din mica experienta obtinuta se pot extrage concluzii privind implicatiile asupra pacientilor, personalului, administratiei, implicatii ce pot avea consecinte considerabile.

Cum principalul scop al SSD este obtinerea rapida si precisa a diagnosticului, intr-o evolutie normala utilizarea SSD determina o imbunatatare a tratamentului pacientului. Astfel cunostintele expertului de inalt nivel si priceperea in judecarea clinica a specialistilor de la nivelele superioare este disponibila pentru pacientii tratati la toate nivelele sistemului de asistenta medicala. Calitatea unui SSD este limitata de utilizarea procedurilor standard. Pacientul poate observa ca procesul consultarii este inuman atunci cind medicul lucreaza cu calculatorul si nu se concentreaza pe relatiile personale si

consideratiile clinice. Daca calculatorul devine o parte integranta a vietii zilnice atitudinea se schimba, in principal prin faptul ca pacientii insisi lucreaza cu acesta. In acest sens sunt disponibile studii de profunzime in Gladys system (Lucas 77). O motivatie serioasa in utilizarea SSD este garantarea stricta a standardizarii principiilor de tratament.

Pentru medic orice formalism pus ca procedura de obtinerea diagnosticului are ca efect cresterea performantelor diagnosticarii. SSD poate asista doctorul in colectarea datelor relevante despre pacientii si produce un bun raport medical. Multe SSD cer informatii standardizate si formalizate ce determina studii prospective si retrospective fata de inregistrările traditionale. Pot fi introduse mai intii astfel de sisteme pentru cresterea aderenței utilizatorilor prin facilitatile ce le ofera in studiile prospective si retrospective. In acest mod SSD poate fi un important instrument in cercetarea clinica din multe zone medicale (Kent 85). Functia de sistem si de motivarea atitudinii medicului SSD poate creste competenta utilizatorului sau reducerea calificarii sale. Daca sunt folosite numai functiile critice sau de invatare, SSD, se constituie intr-un bun instrument de marire a competentei profesionale. Anumiti doctori privesc introducerea SSD in cadrul medical ca un mijloc de coordonare si supraveghere a activitatii lor clinice.

Spitalul poate avea o serie de beneficii prin utilizarea SSD. Un sistem bun poate scadea costurile departamentului dupa cum un sistem cu puncte slabe poate creste numarul testelor de laborator si a investigatiilor. Standardizarea terminologiei creste controlul calitatii cercetarilor clinice. Trebuie considerat si costul achizitiei hardwerului si a sistemului de programe, al scolarizarii personalului. Sunt simplu de estimat influentele timpilor de raspuns, precum si a erorilor ce apar in momentul in care SSD este utilizat in mod curent. Daca SSD necesita pentru utilizare si mentinere personal calificat, sunt necesare cheltuieli substantiale greu de recuperat. Introducerea SSD poate conduce la regindirea structurilor ierarhice traditionale in responsabilitatea deciziei medicale din spitale. O decizie SSD este mai greu de dezbaturat decit o propunere din partea unui membru al colectivului. Daca SSD este utilizat de cadrele din nivelele ierarhice inferioare pot apare conflicte greu de estimat intre acestia si nivelele ierarhice superioare. De asemenea utilizarea SSD in practica medicala va determina a semnificativa alterare a structurii specialistilor. Teoretic utilizarea SSD in spitalele locale va determina reducerea preferintei pacientilor pentru spitalele regionale si universitare ce beneficiaza de personal medical cu inalta specializare.

O serie de aspecte sociale ale utilizarii SSD pe scara larga sint greu de estimat. Utilizarea extensiva a SSD poate determina schimbarea imaginii de zi cu zi a medicilor, multi pacienti vor cere sa participe efectiv la diagnosticare, doctorul fiind mai mult un translator si un comentator. Chiar daca SSD devine disponibil pentru utilizarea in public nu poate prescrie un diagnostic ca o alternativa a consultului medical. Profesiunea

medicală strictă accepta noile metode de decizie și multe reguli impuse de SSD prin diferite autorități medicale naționale și internaționale pot determina dezvoltări majore ale SSD, însă este necesară satisfacerea compromisului între anarhie și sufocare.

Aspectele de natură etică trebuie să fie luate în considerare. Astfel ca este neetic să nu se folosească astfel de sisteme dacă ele conduc la creșterea griii de sănătate. Câteva conflicte etice uzuale în practica medicală pot fi ascunse prin consultarea utilizând SSD. Un exemplu este consultul genetic unde informații biotehnice și statistice pot ocoti întrebări obscure și intime.

Desigur ca se nasc și o serie de alte întrebări în practica utilizării SSD în medicină. Este stabilit principiul ca medicul este responsabil de decizia pe care o ia fața de pacient. În general această responsabilitate se extinde și asupra superiorilor, mai ales asupra șefului de departament. Extensia naturală a acestui principiu face ca decizia sugerată de SSD să confere responsabilitatea în ultima instanță medicului ce a confirmat-o. Astfel, se aliniază cu metodologia practică în care medicul poartă responsabilitatea evaluării sugestiilor critice sau avizării textelor din cărți sau lucrări științifice.

Complexitatea SSD în continua creștere, determină implicații și în alte zone profesionale, cunoașterea de către medici a unor noțiuni legate de știința calculatoarelor, cercetări teoretice ale teoriei probabilității, multilor vagi, cit și implicații provenind din schimbările survenite în structura sistemelor expert. O practică medicală competentă în viitoarele sisteme expert medicale este extinsă la includerea competențelor de înțelegere, certificare și testare. Pe de altă parte erorile ce pot apărea la sistemele computerizate pot cauza multe suferințe, motiv pentru care o legislație specială trebuie să reglementeze acest lucru. Experiența de mai mulți ani în utilizarea SSD arată ca duce la creșterea obligațiilor utilizatorului.

6. CONCLUZII

Desigur ca, punctele de vedere exprimate aici se bazează pe informațiile cuprinse în literatura de specialitate corelate cu observații proprii ale autorului în urma discuțiilor cu personal avizat din domeniul medicinei. Nu au fost tratate detaliile tehnice de construcție a sistemelor suport de decizie cu aplicații medicale, ci doar problematica generală a aplicațiilor cu astfel de destinații.

7. BIBLIOGRAFIE

1. Teza de doctorat, D. Carstoiu 1989.
2. Fuzzy Set Theory and its Applications, H. J. Zimmermann, 1986.
3. Readings in medical artificial intelligence, W. J. Clancey, E. H. Shortliffe, 1990.
4. Logics for artificial intelligence, R. Turner, 1985.

ANALIZA NUMERICA A CIMPULUI MAGNETIC DE EXCITATIE LA UN VIBRATOR ELECTRODINAMIC

Ioan Vadan, Stelian Perde, Anca Farcas
Institutul Politehnic Cluj-Napoca

Rezumat

Se prezinta o analiza numerica a cimpului magnetic de excitatie la un vibrator electrodinamic, utilizand metoda elementului finit implementata pe un calculator PC compatibil IBM, folosind limbajul FORTRAN 77. Programul se poate folosi pentru calculul oricarei configuratii de cimp magnetic stationar plan-parallel sau plan-meridian.

1. Introducere

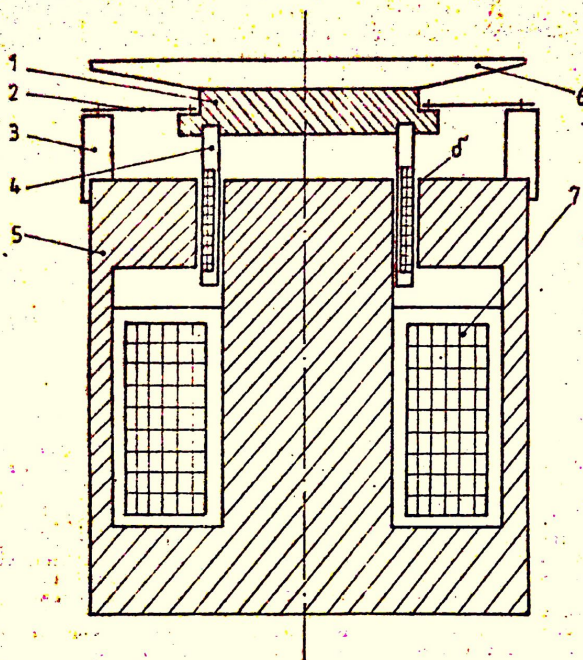
In Figura 1 se prezinta o sectiune printr-un astfel de vibrator electrodinamic. Principiul lui de functionare este asemanator unui difuzor electrodinamic. Bobina de excitatie (7) produce un cimp magnetic constant in intrefierul al circuitului magnetic (5). Aici se afla bobina mobila (4), prin care trece un curent alternativ sinusoidal, care, prin interactiune cu cimpul magnetic constant din intrefier va produce o forta alternativa si sinusoidală care va produce vibrarea echipamentului mobil suspendat de sistemul elastic (2), supunind o eventuala piesa fixata pe platanul (6) la vibratii controlate.

Se pune problema analizei cimpului magnetic constant produs de bobina de excitatie (7) in circuitul magnetic (5).

In prezent, calculul cimpului magnetic la o astfel de constructie de circuit magnetic se realizeaza prin metode numerice [1] in conditii foarte avantajoase, deoarece circuitul magnetic prezinta simetrie cilindrica si calculul cimpului se poate reduce la doua dimensiuni intr-un plan meridian care taie vibratorul prin axa sa.

In lucrare se prezinta solutia aleasa de autori pentru analiza numerica a cimpului magnetic de excitatie la un astfel de vibrator electrodinamic.

S-a adoptat metoda de analiza prin element finit. S-a adoptat solutia elaboratii unui program propriu de analiza numerica a cimpului magnetic fata de posibilitatea achizitionarii unui set de programe de institutii specializate (ex. I.C.P.E. Bucuresti), atat din considerente economice, cit si din posibilitatea creata in acest fel de a dezvolta aceste programe pina la integrarea lor intr-un program complex de proiectare optimizata asistata de calculator a vibratoarelor electrodinamice.



1. Piesă fixare platan bobină mobilă
2. Sistem elastic
3. Suport sistem elastic
4. Bobină mobilă
5. Circuit magnetic
6. Platan
7. Bobină fixă de excitație

Fig. 1.

S-a plecat de la implementari clasice ale metodei elementului finit prin programarea in FORTRAN si rulara pe calculatorul FELIX C-256 utilizand pentru rezolvarea sistemului de ecuatii algebrice subprogramul REBAN din biblioteca matematica MATHLIB.

S-a implementat apoi acest program pe un calculator compatibil IBM-PC AT, utilizand compilatorul F77.

Programul PROF01, este deci scris in limbajul FORTRAN si foloseste date din fisierul FOR01.

In continuare se vor prezenta metoda, programul intocmit si rezultatele obtinute.

2. Metoda elementului finit

In Figura 2 se prezinta discretizarea obtinuta pentru domeniul de calcul ales, domeniu care acopera o semisectiune plan meridiana prin vibrator si un spatiu in aer deasupra intrefierului.

S-au obtinut un numar de 340 de elemente si 198 de noduri. De asemenea, un numar de 48 de elemente sint cu surse, se suprapun peste zona din fereastra circuitului magnetic ocupata de bobina de excitatie. S-a folosit pentru caracterizarea cimpului magnetic potentialul magnetic vector A, care in acest caz cu simetrie cilindrica are doar o singura componenta $A_\varphi = A$.

Considerind cazul cimpului magnetic stationar, in medii fixe, liniare, izotrope, neomogene si nemagnetizate permanent, modelul matematic va fi o ecuatie cu derivate parțiale de tip Poisson :

$$\nabla \cdot (\nabla \bar{A}) = -\bar{J} \quad (1)$$

cu : $\bar{A}$ - potentialul magnetic vector ;
 $\bar{J}$ - vectorul densitatii de curent.

Intr-un cimp bidimensional plan-paralel, ecuatia (1) ia forma :

$$\frac{\partial}{\partial x} \left(\nu \frac{\partial A}{\partial x} \right) + \frac{\partial}{\partial y} \left(\nu \frac{\partial A}{\partial y} \right) = -J \quad (2)$$

Intr-un cimp bidimensional plan-meridian, ecuatia (1) ia forma :

$$\frac{\partial}{\partial r} \left[\frac{\nu}{r} \frac{\partial (rA)}{\partial r} \right] + \frac{\partial}{\partial z} \left(\nu \frac{\partial A}{\partial z} \right) = -J$$

Daca se noteaza cu $\Psi = rA$, o asa-numita functie de flux atunci ecuatia (3) devine

$$\frac{\partial}{\partial r} \left(\frac{\nu}{r} \frac{\partial \Psi}{\partial r} \right) + \frac{\partial}{\partial z} \left(\frac{\nu}{r} \frac{\partial \Psi}{\partial z} \right) = -J$$

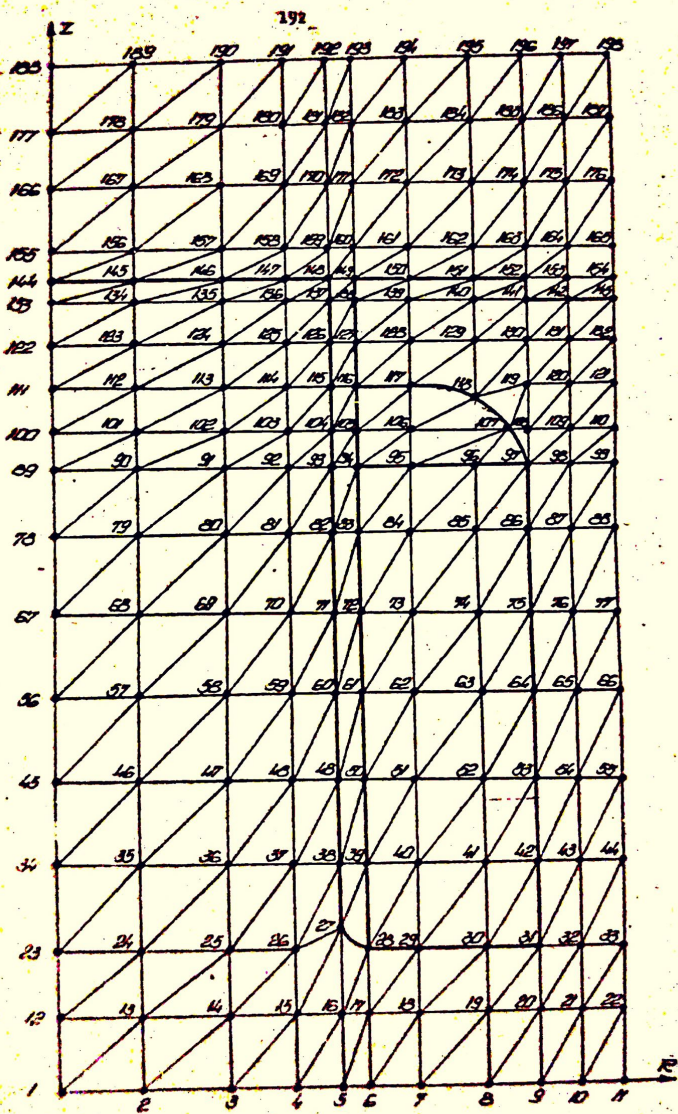


fig. 2

E16

Se observa similitudinea dintre relatia (2) si (4), ceea ce ne permite sa folosim acelasi program, atat pentru analiza cimpurilor magnetice cu simetrie plan-paralela, cit si cu simetrie plan-meridiana prin trecerea de la coordonatele x, y la r, z si de la A la Ψ cu modificari minimale.

Noi consideram cimpul descris de ecuatia (4).

Conditiiile pe frontiera domeniului le alegem de tip Dirichlet, si anume $\Psi|_n = 0$, adica consideram potentialul magnetic vector pe frontiera nul, si prin urmare va fi nul si potentialul magnetic vector modificat $\Psi = r A$.

Considerind o formulare variationala a problemei de cimp descrise de ecuatia (4), se demonstreaza in [1] ca rezolvarea ei este echivalenta cu minimizarea functionalei energetice:

$$F(\Psi) = \int_{\Omega} \left\{ \frac{1}{2r} \left[\left(\frac{\partial \Psi}{\partial z} \right)^2 + \left(\frac{\partial \Psi}{\partial r} \right)^2 \right] - \int \Psi \right\} dz dr - \int \frac{1}{r} \frac{\partial \Psi}{\partial n} \Psi d\Gamma \quad (5)$$

unde Γ este frontiera domeniului Ω .

La nivelul fiecarui element e se aproximeaza functia Ψ prin:

$$\Psi^e(r, z) = \sum_{i=1}^3 \varphi_i \Psi_i = [\Phi]^e [\Psi]^e \quad (6)$$

unde functiile φ_i au expresia:

$$\varphi_i = \frac{a_i + b_i \cdot z + c_i \cdot r}{25^e} \quad (7)$$

iar:

$$a_i = \eta_j \cdot z_k - \eta_k \cdot z_j \quad (8)$$

$$b_i = z_j - z_k \quad (9)$$

$$c_i = \eta_k - \eta_j \quad (10)$$

iar $a_j, a_k, b_j, b_k, c_j, c_k$ au expresii analogice.

Conditia de stationaritate pentru functionala (5) se scrie:

$$\frac{\partial F}{\partial \Psi_i} = 0$$

adica la nivelul unui element:

$$\int_{\Omega_e} \frac{\partial}{\partial \Psi_i} \left\{ \frac{1}{2r} \left[\left(\frac{\partial \Psi_e}{\partial z} \right)^2 + \left(\frac{\partial \Psi_e}{\partial r} \right)^2 \right] \right\} dr dz - \int_{\Gamma_e} \int \frac{\partial \Psi_e}{\partial \Psi_i} dr dz - \int_{\Gamma_e} \left(\frac{1}{r} \frac{\partial \Psi_e}{\partial r} \right) \frac{\partial \Psi_e}{\partial \Psi_i} d\Gamma_e = 0 \quad (11)$$

In ipoteza ca pe un element

$$\Psi = \Psi^e = \text{const.} \quad \text{si} \quad r = r^e = \frac{1}{3} \sum r^e = \text{const.}$$

primul termen din ecuatia (11) devine

$$\frac{1}{r_e} \int_{\Omega_e} \left(\frac{\partial \Psi}{\partial z} \right)^e \left[\frac{\partial \phi}{\partial z} \right]^e \{ \Psi \}^e + \frac{\partial \Psi}{\partial r} \left[\frac{\partial \phi}{\partial r} \right]^e \{ \Psi \}^e \right) dr dz =$$

$$= \frac{1}{r_e(4 \cdot 5e)} \sum_{j=1}^3 (b_i b_2 + c_i c_2) \Psi_j = \sum_{j=1}^3 K_{ij} \Psi_j \quad (12)$$

In mod similar, considerind $J = J^e \text{ const}$, termenul al doilea din ecuatia (11) devine :

$$\int_{\Omega_e} J \frac{\partial \Psi}{\partial \Psi_i} dr dz = J^e \int_{\Omega_e} 1 dr dz = \frac{J^e \cdot S^e}{3} \quad (13)$$

Termenul al treilea din ecuatia (11) in cazul nostru este nul, deoarece nu avem conditii Neumann neomogene pe frontiera $\partial \Omega / \partial n \neq 0$.

Ca urmare ecuatia (11) se scrie matricial

$$K^e \{ \Psi \}^e - \{ F \}^e = 0 \quad (14)$$

cu matricile $K^e (3 \times 3)$ si $F^e (3 \times 1)$ avind componentele

$$K_{ij}^e = \frac{1}{r_e(4 \cdot 5e)} (b_i b_j + c_i c_j) \quad (15)$$

$$f_i^e = \frac{J^e S^e}{3} \quad (16)$$

Scriind ecuatii pentru toate triunghiurile domeniului Ω rezulta un sistem de forma :

$$K \{ \Psi \} - \{ F \} = 0 \quad (17)$$

unde necunoscute sînt valorile Ψ in nodurile retelei de discretizare.

Acesta este un sistem de ecuatii algebrice cu proprietatea ca matricea coeficientilor K este putin umpluta si are o structura de banda.

Acest sistem se poate rezolva cu ajutorul subprogramului REBAN din biblioteca matematica MATHLIB a limbajului FORTRAN. Acest subprogram utilizeaza metoda eliminarii a lui GAUSS pentru rezolvarea sistemelor de ecuatii cu matricea coeficientilor in forma de banda in jurul diagonalei principale.

Daca se tine cont ca aceasta matrice este si simetrica fata de prima diagonala, atunci se poate utiliza pentru rezolvare metoda radacinii patrata sau de eliminare a lui CHOLESKI.

Aceasta metoda este utilizata in programul ce va fi prezentat in continuare.

Matricea K se pune sub forma produsului a doua matrici triunghiulare : $K = L \cdot L^T$ unde

$$L = \begin{bmatrix} l_{11} & 0 & \dots & 0 \\ l_{21} & l_{22} & \dots & \\ \vdots & \vdots & \ddots & \vdots \\ l_{n1} & l_{n2} & \dots & l_{nn} \end{bmatrix} ; \quad L^T = \begin{bmatrix} l_{11} & l_{12} & \dots & l_{1n} \\ 0 & l_{22} & \dots & l_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & l_{nn} \end{bmatrix}$$

si $l_{ij} = l_{ji}$ prin relatia :

$$l_{ii} \cdot \sqrt{K_{ii}} = \sum_{p=1}^{i-1} l_{pi}^2, \quad l_{ij} = \frac{K_{ij} - \sum_{p=1}^{i-1} l_{pi} l_{pj}}{l_{ii}} \quad (18)$$

Ecuația $K \cdot \Psi = F$ devine $L \cdot L^{-1} \cdot \Psi = F$ și notind $L^{-1} \cdot \Psi = Y$ obținem pe Y rezolvind ecuația $L \cdot Y = F$ și apoi pe Ψ rezolvind ecuația $L^{-1} \cdot \Psi = Y$, ceea ce se face ușor deoarece matricile L și L^{-1} sunt triunghiulare. Se va obține

$$Y_i = \frac{F_i - \sum_{p=1}^{i-1} l_{ip} Y_p}{l_{ii}}, \quad i=1, 2, \dots, n \quad (19)$$

și apoi

$$\Psi_i = \frac{Y_i - \sum_{p=i+1}^n l_{ip} \Psi_p}{l_{ii}}, \quad i=n, n-1, \dots, 1 \quad (20)$$

la $i=n$ se ia $\Psi_n = Y_n / l_{nn}$

Dupa rezolvare s-au calculat caracteristicile diferentiale ale cimpului, și anume componentele inducției magnetice cu relațiile:

$$B_r = -\frac{\partial A}{\partial z} = -\frac{1}{r} \frac{\partial \Psi}{\partial z} \quad (21)$$

și

$$B_z = \frac{1}{r} \frac{\partial}{\partial r} (A_r) = \frac{1}{r} \frac{\partial \Psi}{\partial z} \quad (22)$$

3. Programul de calcul a cimpului

În Figura 3 se prezintă organigrama programului de calcul a cimpului magnetic. Acest program se compune din 7 blocuri distincte.

DATE INTRARE, asigură introducerea datelor de intrare cu variabilele mai importante: XT(198), YT(198) coordonatele nodurilor;

LNQDB(340,4) - o matrice care memorează la fiecare element nodurile alăturate lui în sens invers trigonometric și cifra 0 dacă elementul este în aer și 3 dacă elementul este în fier; ea este echivalentă cu mulțimea MNQDB(1360);

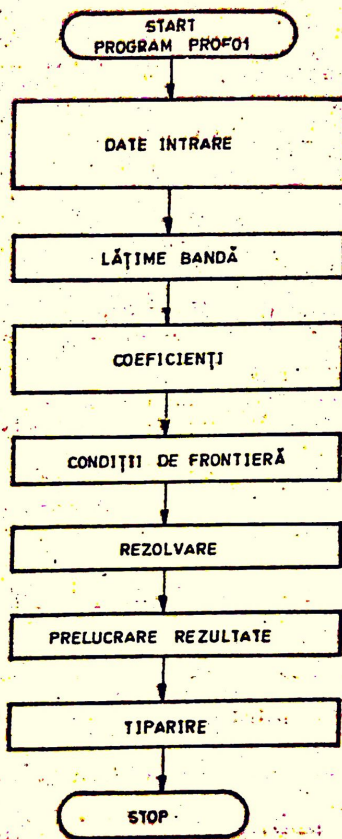


Fig. 3.

E 16

ESRES(3,3), AK(3), BK(3), CK(3) memoreaza caracteristici geometrice ale elementelor ;
 NELEM si MELEM reprezinta numarul de elemente ;
 NPOIN reprezinta numarul de noduri ;
 NVFIX reprezinta numarul de noduri cu potential fix ;
 NPROP reprezinta numarul de elemente cu surse ;
 LVFIX(54) este lista nodurilor cu potential fixat ;
 PVFIX(54) reprezinta potentialele fixe ale nodurilor ;
 PROPEB(48) sint densitatile de curent in elementele cu surse
 NPROPI(48) reprezinta lista elementelor cu surse ;
 ASRES(4500) este multimea elementelor matricei
 coeficientilor ;

ASTOK(198) este matricea coloana a termenilor liberi.

Blocul LATIME BANDA calculeaza latimea benzii si semibenziilor matricei coeficientilor.

Blocul COEFICIENTI calculeaza elementele matricei coeficientilor K si a matricei termenilor liberi a ecuatiei algebrice obtinute dupa formulele deduse la paragraful 2.

Blocul CONDITII DE FRONTIERA asigura modificarea matricei coeficientilor in conformitate cu conditiile de frontiera date. In cazul nostru, la nodurile aflate pe frontiera domeniilor, deoarece potentialul este nul, se vor egala cu 0 toti coeficientii de pe coloana i si linia i care corespund nodului respectiv din matricea K, cu exceptia lui Kii, precum si termenul liber corespunzator, f_i .

Blocul REZOLVARE asigura rezolvarea sistemului de ecuatii algebrice prin metoda radacinii patrata.

Blocul PRELUCRARE REZULTATE asigura obtinerea din potentialele nodurilor a componentelor inducției magnetice B_r si B_z pe fiecare element.

Blocul TIPARIRE asigura tiparirea acestor rezultate.

4. Concluzii

Acest program de calcul de cimp permite calculul cimpului magnetic plan-meridian si cu mici modificari si a celui plan-paralel. Discretizarea domeniului de calcul se face manual si rezultatele se dau sub forma unor liste ale potentialelor magnetice A in noduri si ale inducției magnetice si a componentelor sale in fiecare element. Domeniul de calcul prezentat in Figura 2 este luat dupa un vibrator electrodinamic existent si s-au facut masuratori cu Teslametru a inducției magnetice in intrefier la curenti prin bobina de excitatie egali cu cei considerati in datele din program, obtinindu-se o suprapunere foarte buna a datelor masurate peste cele calculate. In plus, s-a putut constata o corespondenta perfecta a semnelor componentelor inducțiilor magnetice in diferite zone ale circuitului magnetic, in totala concordanta cu cele presupuse teoretic.

Noi consideram acest rezultat deosebit de incurajator si in viitor dorim sa perfectionam acest program, prin completarea lui cu un subprogram de discretizare automata a domeniului de calcul, precum si cu un subprogram pentru trasarea (desenarea) liniilor de cimp. De asemenea, intentionam sa transcriem programul intr-un limbaj de programare mai evoluat, TURBO PASCAL sau TURBO C.

BIBLIOGRAFIE

1. Mindru, Gh., Radulescu, M.M. Analiza numerica a cimpului electromagnetic. Editura "Dacia" Cluj-Napoca, 1986.
2. Demircian, K.S., Ceciurin, V.L., Masinie rasceti electromagnitnih polei, Moskva, Vissiaia Scola, 1986.
3. Cirstea, I., Pasare, S., Un model hibrid pentru cimpul magnetic stationar bidimensional, in E.E.A. Electrotehnica, nr.7/1989.

Metodă de apreciere a comportării întreruptoarelor
de înaltă tensiune la comutație cu ajutorul
calculatoarelor electronice

Ing. Curcanu George - I.U.M.E.T.-Craiova
Prof.dr.ing. Floriganță-Gh.-Univ.din Craiova
Ing. Defta Mihail - I.U.M.E.T.-Craiova

1. Introducere

Aprecierea comportării întreruptoarelor de înaltă tensiune la comutația curenților capacitivi și stabilirea posibilităților limită de comutație a curenților de scurtcircuit se face prin determinarea caracteristicii dielectrice de gol ce a dat rezultate foarte bune pentru întreruptoarele cu aer comprimat și hexafluorură de sulf (1).

Pentru întreruptoarele cu ulei puțin problemele sînt mai complexe, deoarece în acest caz există variații importante ale tensiunii de străpungere, datorate condițiilor mediului de stingere în intervalul creat prin deschiderea contactelor, cît și cele datorate fenomenelor statistice care apar în cazul străpungerilor dielectrice.

În acest caz se determină utilizînd calculatorul electronic, funcția de comportarea întreruptorului pe baza modelului matematic al rigidității dielectrice a intervalului întreruptorului.

2. Experimentări pentru determinarea caracteristicii dielectrice de gol.

Cu instalația realizată în Laboratorul de Mare Putere al ICSITMAE- Craiova s-au determinat tensiunile de străpungere între contactele în curs de deschidere ale întreruptorului cu ulei puțin tip UN - 12 KV/1250 A, pe baza cărora se trasează caracteristica dielectrică de gol.

Schema de principiu a instalației este prezentată în fig.1.

Pe polul T al întreruptorului cu ulei puțin tip UN-12 KV/1250A s-a aplicat tensiunea continuă de la 50 KV în creștere cu 5 KV pînă la 100 KV. S-au efectuat mai multe încercări, astfel că pentru fiecare treaptă a traductorului s-au determinat cel puțin 20 tensiuni de străpungere. Pentru fiecare treaptă a traductorului se consideră parametrii constanți ai intervalului de rupere. Repetind pe un număr mare de ori în condiții identice măsurile tensiunii de străpungere, se constată, în general, că rezultatele aleatorii măsurărilor sînt caracterizate de o lege de repartiție bine determinată. În acest caz se poate determina valoarea medie μ și dispersia σ .

./.

acestei repartiții normale (Gauss) [1,2].

Fie $u_1, \dots, u_n$ tensiunile de străpungere corespunzătoare unei trepte a traductorului. În acest caz:

$$\mu = \frac{\sum u_j}{n} ; \quad \sigma = \sqrt{\frac{\sum u_j^2 - \mu^2}{n-1}} \quad (1)$$

cunoscându-se aceste date, rezultă funcția de comportare [2]:

$$S_i(u) = \frac{1}{\sigma_i \sqrt{2\pi}} \int_{-\infty}^u e^{-\frac{(t-\mu_i)^2}{2\sigma_i^2}} dt \quad (2)$$

Pentru comportarea întregului întreruptor se scrie funcția cumulativă:

$$S(u) = \sum_{j=1}^K \prod_{L=1}^j [1 - S_{i-1}(u)] \cdot S_i(u) \quad (3)$$

care determină probabilitatea de străpungere în timpul intervalului K sau în cursul intervalului precedent.

K - reprezintă numărul de trepte ale traductorului de cursă

3. Programe realizate pe calculator pentru determinarea probabilității de străpungere

Pe calculatorul Independent I-102 F din dotarea Laboratorului de Mare Putere, s-au realizat două programe de calcul în limbaj FORTRAN pentru determinarea probabilității de străpungere a întreruptorului între contactele în curs de deschidere ale întreruptorului cu ulei puțin.

- Programul a fost realizat pentru :
 - verificarea normalității șirului de date experimentale;
 - calculul valorii medii și al dispersiei
- Programul II a fost realizat pentru:
 - calculul funcției de comportare: K . S_i ;
 - calculul funcției cumulative: $S(u)$.

Programul I

Există un număr m de date experimentale care reprezintă tensiunile u_1 , de străpungere a intervalului, obținute în urma experimentărilor efectuate cu instalația laboratorului.

Măsurătorile au fost repetate cu instalația laboratorului repetate de mai multe ori și s-a constatat că rezultatele aleatorii u_1 , sînt caracterizate de o lege de repartiție normală.

În acest scop s-a verificat normalitatea șirului de date experimentale pentru fiecare treaptă la care s-a inițiat străpungerea. Verificarea normalității șirului de date experimentale s-a făcut pe baza metodei diferențelor succesive a lui E-Abbe (6).

Testul lui R. Abbe constă în următoarele:

- se introduc datele într-un fișier pe disc;
- se ordonează datele girului în ordine crescătoare:

$$u_1 < u_2 < \dots < u_n;$$

- se calculează valoarea medie μ și dispersia σ^2 cu relațiile:

$$\mu = \frac{1}{m} \sum_{j=1}^m u_j; \quad \sigma^2 = \frac{\sum_{j=1}^m (u_j - \mu)^2}{m-1} \quad (4)$$

- se calculează raportul γ cu relația:

$$\gamma = \frac{\sum_{j=1}^m (u_{j+1} - u_j)^2}{\sum_{j=1}^m (u_j - \mu)^2} \quad (5)$$

- se alege pragul de semnificație $q = 0,05$;

- se calculează γ_q cu relația:

$$\gamma_q = 1 - \frac{1,645 \sqrt{m-2}}{\sqrt{(m-1)(m+1)}} \quad (6)$$

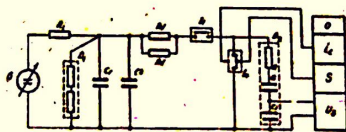


Fig. 1. Schema de principiu a instalației pentru determinarea tensiunii de străpungere a intervalului întreruptorului; G — sursă de tensiune continuă variabilă; R_1 — rezistență de protecție a sursei; D_1 — divizor rezistiv pentru măsurarea tensiunii sursei; C_1 ; C_2 — condensatoare de înaltă tensiune; R_2 — rezistență de descărcare; I_1 — întreruptor auxiliar; I_2 — întreruptor de încercat; D_2 — divizor capacitiv pentru măsurarea tensiunii de străpungere; C_3 ; C_4 — condensatoare de înaltă și joasă tensiune; O — oscilograf; I_c — curent comandă aparat; S — cură traductor; U — tensiune de străpungere.

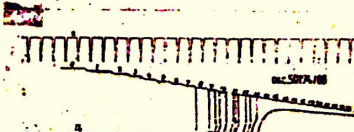


Fig. 2. Oscilograma tensiunii de străpungere între contactele la care se deschide întreruptorul tip LN-13 kV/1250 A.

- unde m reprezintă numărul de date experimentale.
 Dacă se constată că $\frac{s}{\sigma_0} \leq 1$, se admite ipoteza normalității
 șirului de date. Organigrama programului I este dată în fig.3.
 Conforma acestui program rezultă normalitatea șirului de date.

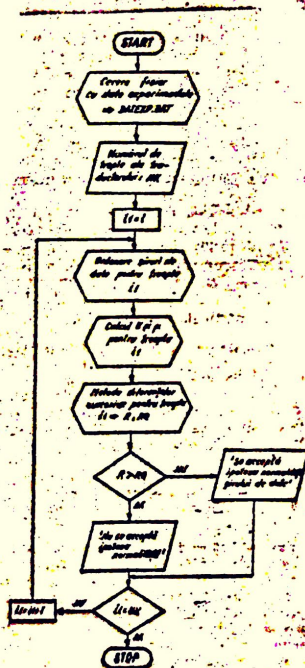


Fig. 3. Organigrama programului I de calcul.



Fig. 4. Organigram programului II de calcul.

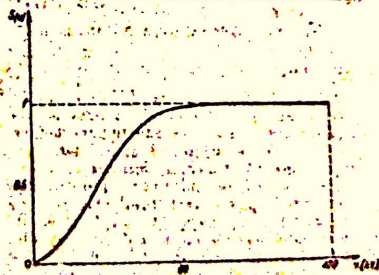


Fig. 5. Graficul funcției cumulative transată pe calculator.

Programul II

Se calculează K_5 -funcție de comportare și se trasează grafic curbele funcției de comportare pentru fiecare treaptă.

S-a calculat de asemenea, funcția cumulativă $S(u)$, care dă probabilitatea de străpungere a intervalului și s-a trasat curba funcției cumulative. Organigrama programului de calcul este prezentă în fig.4, iar graficul funcției cumulative este dat în fig.5.

4/ Concluzii

Pentru aprecierea comportării întreruptoarelor de medie tensiune cu ulei puțin la probele de comutație se procedează astfel:

- se determină tensiunile de străpungere ale intervalului dintre contactele în curs de deschidere ale întreruptorului în stare nouă de funcționare;
- se ordonează șirul de date experimentale și se verifică dacă se supun unei repartiții normale;
- se determină funcția de comportare a întreruptorului utilizând modelul matematic al rigidității dielectrice a intervalului întreruptorului.

Pe baza acestei metode se poate estima mai exact probabilitatea de defect a întreruptorului, decît prin trasarea directă a caracteristicii de gol a întreruptorului.

Bibliografie

1. Fröhlich, K. Mesure des caractéristiques de tension et performances de coupure des disjoncteurs. In: CIGRE-13.01.1978
2. Elberberger, H. Statistische Verfahren zur Bestimmung der Durchschlagwahrscheinlichkeit von Isolierstrecken. In: Wiss. Z. Elektrotech. Leipzig 17, 1971, 2/3 S.- 11713-2.
3. Curcanu, G.; Rărinca, O.; Burlacu, I. Equipment for determination of the voltage circuit-breaker dielectric rigidity characteristic. In: Technical reports-Electroputere Craiova-1986
4. Curcanu, G., Rărinca, O., Defta, M. Raesuzhdenia nad harakteristikami holostogo hoda visokovoltnih vikliuchatelej. Conferinta de 40-rtaj Electric- INTERELEKTRO, Togliatti, URSS, 12-17-09-1989.
5. Curcanu, G., Defta, M., Müller, M. Vozmojnosti ispolzovania dielektricheskijn karakteristik holostogj hoda pri prognozirovanii povedenia maslozbiomnih masliannih vikliuchatelej na srednem napriazhenii. In: Referatele Conferinței Jubiliare a IET-Sofia, 27-28-10-1988.
6. Tiron, M. Teoria erorilor de măsurare și metoda celor mai mici pătrate. Editura Tehnică, București 1972.

PROGRAM PENTRU CALCULUL NECESARULUI DE CALDURA PENTRU CONSTRUCTII CIVILE SI INDUSTRIALE

autorii: conf.dr.ing.Claudia Botez, I.P.Iași
ing.Virgil Botez, Camera de Comerț Iași

Definirea necesarului de căldură pentru clădirile civile și industriale este o etapă premergătoare proiectării și dimensionării instalațiilor de încălzire.

Procedura de calcul a necesarului de căldură este reglementată de prevederile STAS 1907/ 1 și 2 din 1980, STAS 6472/ 1, 2 și 3 din 1989 completate de instrucțiunile 41 din 18.12.89 ale ICCPDC cu privire la asigurarea stabilității termice.

Luorarea își propune să prezinte o procedură de calcul a necesarului de căldură, elaborată pentru calculatoare personale compatibile IBM - PC.

Procedura nu se aplică la: încăperi subterane, clădiri climatizate, clădiri cu destinații speciale.

1. Prezentare generală

În fig.1 este dată schema generală de calcul din care se desprind două etape de lucru:

- calculul coeficienților de masivitate termică al elementelor de construcție (conform STAS 6472/3 din 1989)
- calculul necesarului de căldură (conform STAS 1907/1 din 1980).

2. Descrierea procedurii de calcul

Coeficienții de masivitate termică sînt calculați după relațiile indicate în [4] pentru fiecare element de construcție în parte. Acești coeficienți se depun în tabele care trebuiesc actualizate pentru fiecare construcție-obiectiv în parte.

Pentru calculul necesarului de căldură datele de intrare privesc construcția în ansamblu, încăperea și elementele de construcție ale unei încăperi.

Necesarul de căldură corespunde pierderilor fiecărui

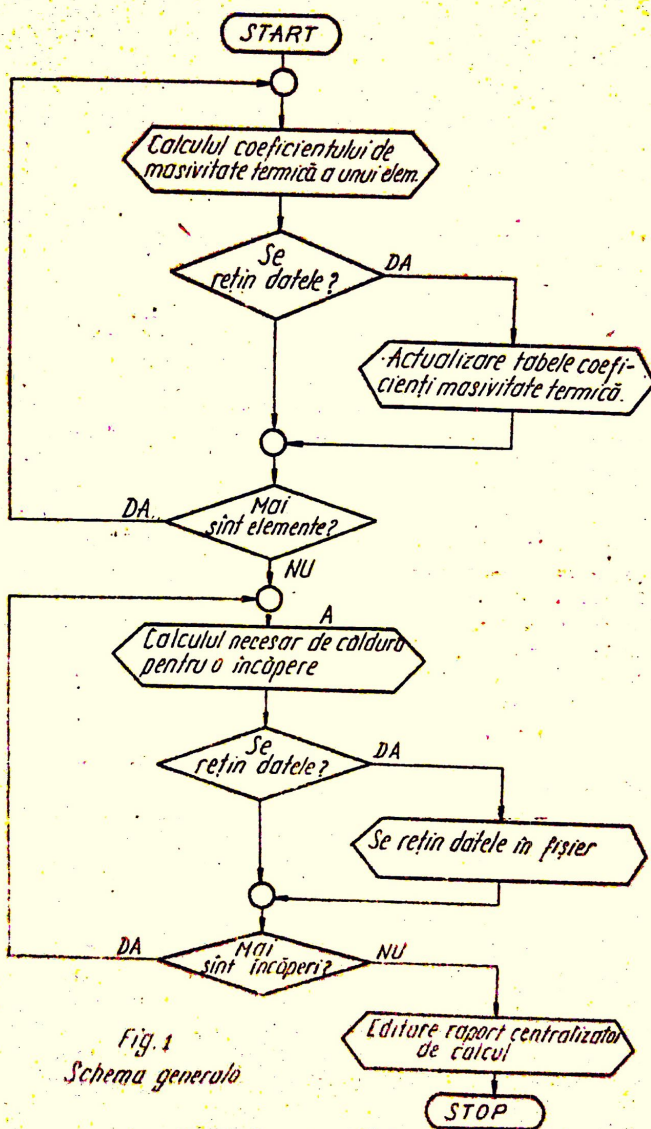


Fig. 1
Schema generală

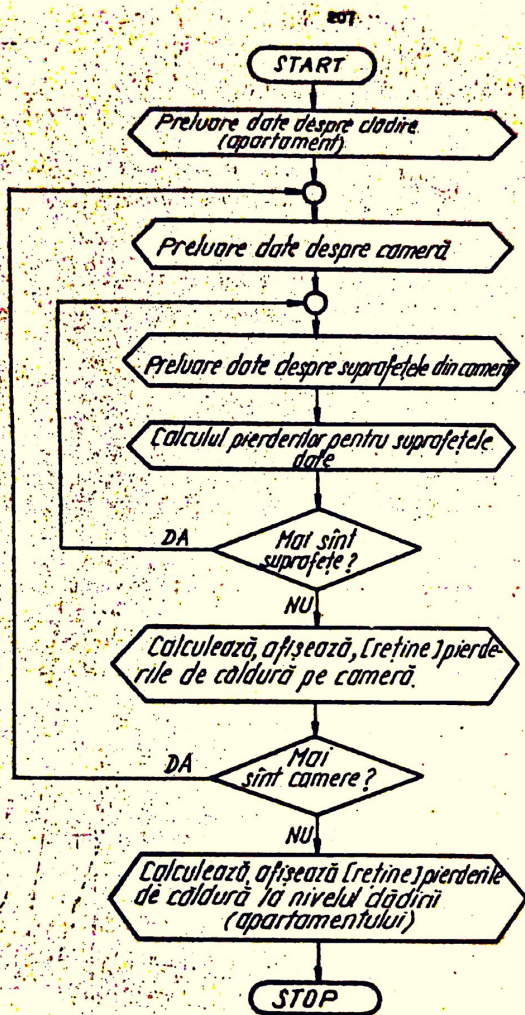


Fig. 2
Calculul necesarului de căldură pentru o clădire (apartament)

fiecărui element de construcție, cumulat la nivelul încăperii, utilizând relațiile de calcul din [2]. Procedura de calcul care realizează această operație este notată cu A în schema generală din fig.1. Schema logică de principiu a acestei proceduri este prezentată în fig.2.

Programele sînt scrise în C, putînd fi rulate pe un calculator compatibil IBM. Datorită numărului relativ mare de date de intrare pe care trebuie să le introducă utilizatorul, s-a urmărit proiectarea unei interfețe om-măgînă cît mai prietenoasă. Astfel pot fi alese culorile ecranului și a textului, după preferință. Mesajele atenționează asupra datelor care trebuie citite, precizîndu-se natura lor și lungimea maximă a șirului care va fi citit, în invers video. Imediat după citire, fiecare dată de intrare este validată sau nu, după cum este sau nu de tipul așteptat. Dacă data introdusă este eronată, se reia mesajul, pînă cînd data este corectă.

De asemenea după un grup de operații, utilizatorul este chestionat asupra corectitudinii rezultatelor. Dacă utilizatorul consideră că ele sînt corecte, etapa se reia.

Programul propus a fost testat cu rezultate bune în calculul necesarului de încălzire într-un atelier școală, într-o hală industrială și în încăperile anexe acestora.

BIBLIOGRAFIE

1. IBM Turbo C - Manual de utilizare
2. STAS 1907/1 - 80 Calculul necesarului de încălzire
3. STAS 1907/2 - 80 Calculul necesarului de încălzire
4. STAS 6472/3 - 89 Calculul termotehnic al elementelor de construcție ale clădirilor
5. ICGPDC - Instrucțiuni tehnice pentru proiectarea la stabilitate a elementelor de închidere a clădirilor - Buletinul construcțiilor nr.1 din 1990.

Sistem informatic pentru urmărirea producției într-o secție de țesătorie sau finisaj

Autori: c.p. III R. Romanescu, ing. C. Codreanu

ing. C. Moșu I.P.A. Iași

as. ing. A. Condorevici, I.P. Iași

Acest sistem informatic de proces realizează culegerea automată și prelucrarea în timp real a datelor privitoare la situația unei secții de țesătorie în ceea ce privește starea de funcționare, gradul de încărcare al mașinilor, lucrătorilor, schimburilor, volumul global al producției sau cel realizat pe diverse perioade de timp, tipuri de produse finite, tipuri de mașini, pontajul automat, etc.

Introducere

O secție de țesătorie este formată din cîteva sute de mașini de țesut. Fiecare mașină de țesut este prevăzută cu un contor mecanic, care contorizează numărul de rotații al arborelui principal.

Numărul de mișcări bătăi, pe care îl conține nota cu "n", constituie un element principal în estimarea producției de material țesut. Cu ajutorul acestei mărimi, și cunoscînd constanta specifică a materialului, se poate afla cantitatea de material produsă pe durata unui anumit interval de timp.

Pînă acum, sistemul de urmărire a producției se baza pe preluarea manuală a datelor de pe fiecare mașină, respectiv "n" bătăi, citite pe contorul propriu al mașinii în sistemul țesuturii achiziționate de către conducătorul întreprinderii.

Acesta mai are obligația de a consemna și stagiul de funcționare-staționare al mașinii. Toate aceste date se completează într-o fișă de lucru a mașinii. Aceste fișe se centralizează la nivel de secție pe schimburi, zile și luni și se prelucrau semiautomat de către o grupă economică din cadrul secției.

Lucrarea s-a materializat în cadrul unui proiect de sistem

sistematiză și informatică care are ca scop trecerea evidenței primare din cadrul secției de testări la prelucrarea și evidența pe calculator a tuturor datelor necesare urmăririi în timp real a producției. Aceasta va avea ca rezultat mărirea flexibilității în conducere și o mai mare mare valoare decizională.

Se urmăresc și realizează:

- * cuantificarea fiecărei mașini în cadrul sistemului informatic;
- * înregistrarea timpului de funcționare și staționare a fiecărei mașini; mașini;
- * în cadrul timpilor de staționare, aceștia se vor explicita corespunzător cauzei ce i-a produs. Se vor avea în vedere următoarele cauze:
 - reparații planificate, reglare rășboare, lipsă materie primă, lipsă piese auxiliare, lipsă lucrător;
 - reparații accidentale (semnalizate de pe fiecare mașină în parte); parte;
 - lipsă energie.

Descrierea funcțională

Echipamentul de prelucrare va fi organizat pe două nivele:

- a) calculatorul de proces;
- b) interfețele specifice mașinilor și interfața calculatorului cu procesul.

Schea bloc a întregului ansamblu poate fi urmăriată în fig.1.

Se disting următoarele blocuri funcționale:

- calculatorul de proces (1);
- interfața specifică a acestuia (2);
- linie de transmisie (3);
- interfețe specifice mașinilor (4);
- bloc alimentare (5) BA.

Interfața specifică a calculatorului realizează adaptarea între sistemul de calcul și linia de transmisie din punct de vedere fizic și al compatibilității nivelelor logice. Ea are drept scop realizarea protocolului de emisie-recepție a

calculatorului cu restul instalației.

Linia de transmisie realizează legătura fizică între interfața specifică a calculatorului și interfețele specifice mașină. Interfețele specifice mașină realizează preluarea locală și memorarea datelor și protocolul de emisie-recepție cu calculatorul.

Modul de funcționare este următorul: calculatorul interoghează la un anumit moment o singură interfață specifică mașină prin emisia adresei acesteia. Interfețele specifice mașină sînt în mod normal în starea de recepție. Interfața care își recunoaște adresa va trece ulterior în stare de emisie. Ea va transmite către interfața specifică a calculatorului (deci unitatea centrală) valoarea lui "M" reprezentînd numărul de mișcări bătăi, înregistrat în acel moment pe mașină, precum și semnalele de stare ale mașinii.

Preluarea datelor se va face de două ori consecutiv pentru fiecare mașină, pentru a se elimina o parte din posibilele erori apărute la preluarea primară a datelor.

Descrierea constructivă

Interfața specifică mașină

Această parte a instalației se află amplasată în apropierea mașinilor supravegheate. Pentru a proteja montajul se recomandă montarea alăturată de mașină, nu pe șasiul acesteia (supus unui regim considerabil de vibrații).

Montajul este realizat constructiv pe o monoplață de cablaj imprimat, conținînd un sistem dedicat, cu microprocesor. Două capace fixate cu șuruburi permit accesul la ambele fețe ale cablajului. Prin bușe izoante, fixate în ramă sînt introduse trei cabluri:

- a) cablul de alimentare;
- b) linia de transmisie;
- c) racordul la traductoarele de pe mașină (realizat cu cablu ecranat).

Linia de transmisie va fi echipată corespunzător condițiilor concrete de funcționare.

Se recomandă montarea direct pe această cutie a comutatoarelor de stare (oprire, avarie-reparație accidentală)

precum și a semnalizării optice (optională) aferente: alimentare interfață, maritori ai pozițiilor comutatoarelor, etc.

Interfața specifică calculator

Această parte a sistemului, spre deosebire de informația specifică mașină, este unică și funcționează în condiții mult mai ușoare, alături de calculator.

Din punct de vedere constructiv, este realizată pe cablaj imprimat, montat într-o casetă.

Linia de transmisie

Această parte a sistemului are rolul de a asigura calea de comunicație între diversele părți componente. Deși din modul de alegere a protocolului de comunicație s-a urmărit reducerea riscurilor de apariție a erorilor la transmisie, se recomandă alegerea unui cablu ecranat.

Sistemul informatic de proces

Întregul volum de date privind producția de pe fiecare mașină, respectiv starea acesteia, era preluat și prelucrat manual. Aceasta duce la o puternică implicare a factorului uman, care, pe lângă viteza redusă de operare, este și generator al distorsiunii în procesul de achiziție și prelucrare a datelor. Astfel apar de multe ori diferențe semnificative între valoarea producției indicată de contoarele de pe mașină și cea reală. La toate acestea se adaugă și faptul că în secția de producție se prelucrează materii prime de calitate diferite, se produce un sortiment larg de țesături pe baza aceluși proces tehnologic, iar cererile și livrările de produse fluctuează de la o zi la alta.

Întrucât un ține cont doar de aceste considerente, se remarcă oportunitatea introducerii sistemului informatic pentru urmărirea producției într-o întreprindere textilă.

Trebuie menționate în același timp și avantajele ce apar ca urmare a acestei introduceri, avantaje ce în sistemul actual existent, de culegere și prelucrare manuală a informațiilor privind volumul de materii prime, a stării

mașinilor, a volumului producției pe schimburi, etc. nu sînt posibile. Este vorba de o informare operativă și obiectivă a factorilor umani de decizie, precum și a unor eventuale sisteme informatice de gestiune economică de la nivelele terarhice superioare, prin elaborarea periodică sau la cerere a unor rapoarte sau informații adecvate asupra realizărilor sau stării procesului industrial (bilanșuri, situații pe ore, schimburi, zile, etc.).

Trebuie remarcat faptul că sistemul informatic de urmărire a producției poate furniza informații în timp real despre starea secției, permițînd luarea rapidă a unor decizii pentru optimizarea la maximum a procesului de producție. Niciodată nu se va putea ajunge la astfel de performanțe folosînd operatorii umani care să completeze fișele fiecărei mașini, să le centralizeze, să prelucreză informațiile rezultate din aceste fișe și să le prezinte factorului decizional în timp util, ci se va crea întotdeauna un decalaj între starea reală a secției și situația cunoscută de șeful ei.

În această ordine de idei, a avantajelor nou create, prin introducerea sistemului informatic, se înscrisă și posibilitatea unei mai profunde cunoașteri a mecanismelor intime ale procesului industrial prin culegerea și prelucrarea unor informații precise și la timp, ceea ce permite o mai bună încărcare a utilajelor și instalațiilor, o mai bună organizare a muncii, perfecționarea modelelor și algoritmilor de conducere, etc.

Tot acest sistem informatic va mai permite evitarea apariției unor deranjamente sau avarii și limitarea consecințelor acestora, datorită posibilității investigației mai profunde a dinamicii procesului tehnologic, în special în perioadele de regimuri tranzitorii, de perturbații mari și pe baza rapoartelor post-avarie și a analizelor structurate elaborate automat.

Concluzii

Acest sistem informatic de proces, ce poate fi aplicat în unități productive de alt profil, conferă o mai mare activitate și o mai mare viteză decizională, la un cost avantajos (rezultat al unei proiectări inginerie).

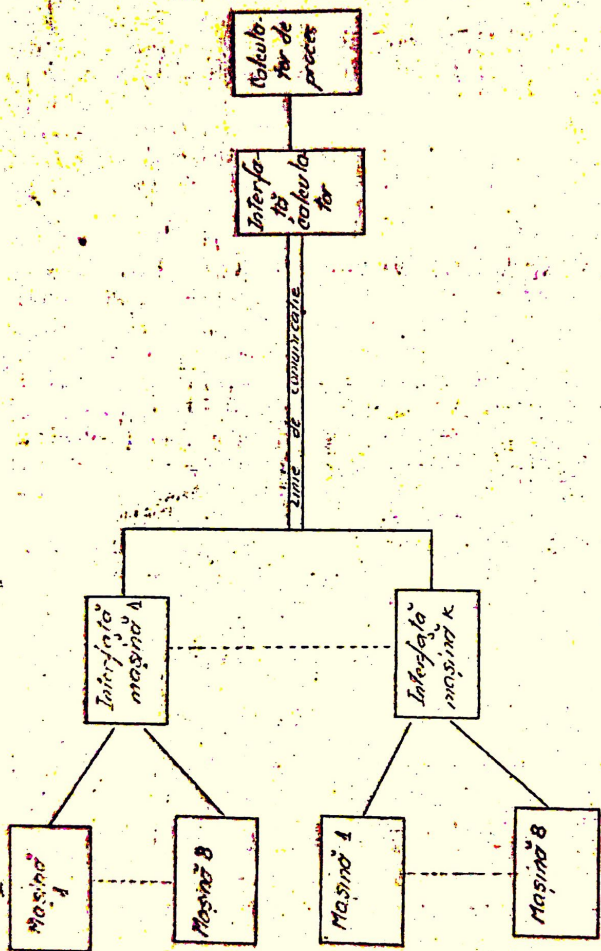


Fig. nr. 1

Sistem informatic pentru urmărirea producției într-o secție de țevărie sau finisaj

SISTEM INFORMATIC PENTRU PROCESE TEHNLOGICE CU ELEMENTE NUMEROASE SI TIPODIMENSIUNI PUTINE

Ing. Costescu-Gh. C-tin-Vasile
 Ing. Georgescu, Cristian
 Universitatea "Dunarea de Jos" - Galati
 Ing. Gheorghe Gabriel
 Ing. Stavila Sorin
 F.E.A. - Bucuresti
 Ing. Vasiliu H. Zonel
 R.E.N.E.L. - F.R.E. Galati

Rezumat

Lucrarea incearca sa raspunda unor gresuti de patrundere a informaticii in domeniul proceselor tehnologice, gresuti ridicate de faptul ca o larga gama de procese au foarte multe elemente de urmarit sau de comandat. Aceste elemente sint de citeva tipodimensiuni, fapt care produce gresuti de realizare hardware si software. In activitatea Filialelor de Retele Electrice din tara se intilnesc in mod curent astfel de situatii. Ideea de baza este automatizarea modulelor de intrari/iesiri numerice in/din calculator, prin preluarea unor functii ale unitatii centrale si amplasarea lor in teren, la distanta de acesta. Se obtine in acest fel o crestere substantiala a fiabilitatii si o scadere de aproximativ 10 ori a costului per semnalizare.

1. Introducere

In procesele tehnologice din ramura electroenergetica, exista foarte multe elemente de acelasi tip sau de citeva tipuri, respectiv aparatura de comutatie primara si/sau secundara, LED-uri sau lampi in panouri sinoptice, etc. Pentru controlul lor se pot folosi microcalculatoare de proces din gamele SPOT 80 si/sau ECAROM-881M, dar, in acest caz, controlul a circa 4000 elemente produce repede fenomenul de "saturare" a modulelor IN si/sau EN, necesitind cheltuieli mari (circa 1000 lei/element) fara a avea asigurata fiabilitatea necesara. Mai este faptul ca, nu se stie cind anume s-a defectat "controlul" unui element oarecare, astfel ca sistemul este si costisitor, si inefficient. S-a impus deci gasirea unui sistem ieftin (100 lei/element) si efficient, in sensul ca este fiabil si totodata capabil sa semnaleze el insusi cind s-a pierdut controlul asupra unui element. Acest sistem se preteaza la aplicarea in electroenergetica la controlul aparaturii de comutatie primara si secundara, panouri sinoptice, etc., unde elementele sint numeroase si de tipuri putine.

2. Arhitectura sistemului

Principiile care stau la baza arhitecturii sistemului sint in numar de trei:

- prelucrarea distribuita pfin "spargerea" microcalculatorului in module amplasate in diverse zone ale procesului tehnologic si nu concentrate intr-un echipament unitar, legatura facindu-se la orice distanta cu sau fara MODEM;

- adresarea elementelor nu prin propria intrare sau iesire numerică, ci prin magistrale de proces separate, acționând câte un pachet de porturi, ori câte un pachet de module;
- evitarea necesității vitezei mari de lucru prin mesaje cu frecvență mică.

Arhitectura a rezultat de la sine după definirea principilor, respectiv un modul central, conectat prin seriialele sale în rețea de calculatoare și la propria-i periferie, modul central legat la modulele de lucru amplasate în instalații prin cabluri jucând rol de magistrală. Aceste cabluri constituie o mică fracțiune din volumul cablurilor soluției tradiționale. Fiecare modul de lucru deservește un subansamblu de elemente, structurat pe considerente de vecinătate a elementelor sau pe considerente de grup funcțional al elementelor.

Modulul central are propria lui periferie și este conectat într-o rețea de calculatoare, de la care primește comenzi, care îi furnizează informații inclusiv despre defectele de control. El are posibilitatea de a lucra și individual, fie ca așa a fost conceput, fie ca este vorba de o avarie.

3. Implicatii și rezolvări software

Funcționarea sistemului este concepută astfel încât microcalculatorul să poată fi integrat într-o rețea, lui revenindu-i sarcina de interfata între procesul tehnologic și sistemele de calcul. De aici au rezultat cele trei module software, respectiv: modulul pentru controlul elementelor de proces, modulul de comunicație cu sistemele externe și modulul de lucru în regim autonom.

Modulul pentru controlul elementelor de proces este realizat sub forma unei imagini-obiect, rămânând definitiv și conceput de fabricant.

Modulul de comunicație este realizat sub forma unui program generator de imagine-obiect, conform cu protocolul sistemului de calcul exterior.

Modulul de lucru autonom este conceput de beneficiar pe baza intrărilor/ieșirilor precizate de foaia de însoțire a echipamentului. Cu ajutorul unui fisier de comenzi indirecte, cele trei imagini-obiect sînt legate, realizîndu-se imaginea executabilă. Beneficiarului nu-i rămîne decît a ține cont de sistemul de calcul exterior, iar pentru modulul autonom de lucru, de modalitatea în care a legat modulele periferice la elementele de proces, respectînd adresa fiecărui element de proces și natura legăturii, (intrare sau ieșire).

4. Realizare practică

Experimentarea s-a făcut la F.R.E. Galați. Ea constă în conectarea modului central la un minicalculator tip CORAL 4030, care îi comunică toate comenzile necesare. Modulul central, amplasat lîngă minicalculator, este legat la un modul de lucru amplasat la distanța de 50 m în panoul sinoptic, executînd comenzi asupra LED-urilor dintr-un subpanou. Lucrarea este în curs de dezvoltare, urmînd ca două panouri sinoptice, aflate în două camere de comandă diferite, fiecare panou avînd cîte 9 subpanouri, cu un total de cîrca 4000 LED-uri, să fie comandate de un singur modul central.

Acesta este amplasat lîngă minicalculator și comanda module de lucru realizate sub forma unor plăci de marimea unui format

A4, montate câte 2-3 bucăți pe spatele ușilor fiecărui subpanou. Legătura se realizează prin trei cabluri cu câte 8 fire, fiecare din ele deservind câte un grup de module de lucru.

Echipamentul execută comenzile asupra LED-urilor, sonerilor și hupelor, efectuând schimbări de stare, pilpiiri cu frecvențe diferite și primind comenzi manuale de la utilizator, de confirmare a pilpiirilor și anulare a sonerilor sau a hupelor.

5. Concluzii

În ramura electroenergetică există activități care, fără a necesita prelucrări de calcul complicate, "salvează" echipamentele prin numărul enorm de elemente de proces. În consecință, s-a impus necesitatea realizării unui microcalculator dedicat, care să joace rol de interfață între elementele de proces și sistemele de control.

S-a avut în vedere, pe de o parte, concepția de realizare pe baza prelucrării distribuite a datelor și adresarea prin magistrale de proces separate cu viteză de lucru obișnuită. Pe de altă parte, s-a avut în vedere implementarea ușoară hardware amplasând descentralizat modulele de lucru și identificând elementele de proces prin adresele intrărilor / ieșirilor. Din punct de vedere software, se realizează o implementare ușoară prin transparența modului de control al elementelor de proces, generarea ad-hoc a modului de comunicație în conformitate cu protocolul utilizatorului și prin posibilitatea unui modul de funcționare în regim autonom.

Costul controlului unui element de proces s-a micșorat de la 1000 lei/element la 100 lei/element.

Bibliografie

- [1] Manual de exploatare și întreținere ECAROM - 881M Editat de F.E.A. - București
- [2] Murray, W. D., Computer and Digital System Architecture, Prentice Hall, N.J. 1990

ESMUL - PACHET DE PROGRAME UTILIZAT PENTRU ESTIMAREA IN TIMP REAL A PROCESELOR MULTIVARIABILE

Ing.D.I.Curiao, Dr.ing.O.Proștan, Dr.ing.B.Lupșean
Universitatea Tehnică Timișoara

1. Introducere

Estimarea proceselor multivariabile este un domeniu de mare actualitate în automatica modernă, stînd la baza analizei și a conducerii adaptive a sistemelor dinamice. Dezvoltarea sistemelor de calcul și a limbajelor evoluata a dat posibilitatea extinderii ariei de utilizare a algoritmilor de estimare multivariabilă la procese rapide.

Lucrarea de față prezintă pachetul de programe ESMUL utilizat pentru estimarea în timp real a parametrilor proceselor multivariabile. Estimarea multivariabilă, ca etapă decisivă în stabilirea performanțelor conducerii adaptive presupune, indiferent de algoritmul utilizat, operații matriciale care necesită un timp de calcul relativ mare. Aceasta depinde în principal de patru factori, dintre care doi "obiectivi" - ordinul procesului tehnic precum și numărul intrărilor și ieșirilor procesului - și doi "subiectivi" - algoritmul de estimare utilizat precum și experiența programatorului. Utilizarea limbajului "C" a facilitat creșterea performanțelor pachetului de programe propus, asigurînd posibilitatea conducerii în timp real a proceselor avînd constante de timp limitate inferior la 1 ms.

2. Descrierea pachetului de programe ESMUL

Pachetul ESMUL are la bază algoritmul de estimare multivariabilă GMP on-line /1/, putînd fi utilizat în trei variante ce se deosebesc prin modul de obținere a seturilor de date intrare-ieșire.

În primele două variante, destinate problemelor de analiză, datele sînt generate prin simularea procesului pe baza modelului său discretizat și respectiv sînt introduse de utilizator de la tastatură. În cea de a treia variantă, destinată comanda-rii în timp real, datele sînt preluate direct din proces.

Pachetul ESMUL, scris în limbajul "C", este user-friendly, operatorul avînd acces la un fișier help ce descrie funcționarea fiecărui program din componență. Este prevăzută totodată posibilitatea reprezentării grafice a parametrilor estimați precum și calculul unor indicatori ai calității estimării.

În prima variantă (date simulate) pachetul ESMUL are structura, din figura de mai jos

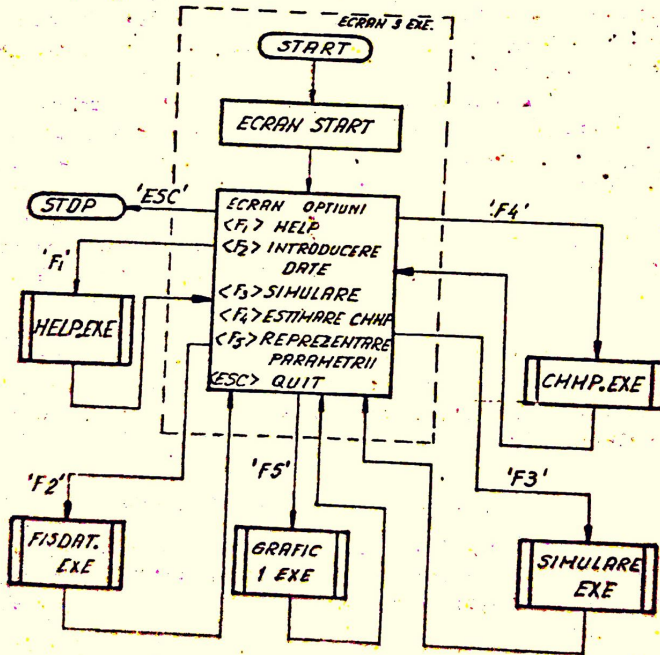


Figura 1. Structura pachetului ESMUL

Rolul central în cadrul pachetului de programe **EXMUL** este deținut de programul **monitor ECRAN3.EXE**, care realizează interfața cu utilizatorul și monitorizarea celorlalte programe. În funcție de opțiunea utilizatorului sau în funcție de un sistem de priorități definit de către acesta, pot fi apelate programele. **HELP.EXE**, **FISDAT.EXE**, **SIMULARE.EXE**, **COMP.EXE**, **GRAFIC1.EXE**, respectiv abandonarea execuției programului și ieșirea în sistemul de operare DOS.

Programul **HELP.EXE** apelează prin apăsarea tastei **F1** afișează pe ecran o serie de informații referitoare la integral pachet de programe, cum ar fi: rolul programelor în cadrul sistemului, inițializarea pachetului de programe, prezentarea algoritmului de estimare multivariabilă utilizat, etc.

Programul **FISDAT.EXE** permite introducerea în fișierul **ordin.dat** a numărului de intrări și ieșiri, a ordinului mediei alunecătoare precum și a ordinului autoregresiei. Fișierul conține numere întregi și este utilizat de programul **COMP.EXE**. La cererea utilizatorului acest program mai permite introducerea seturilor de date intrare-ieșire de la tastatură în fișierul **inties.dat**.

Programul **SIMULARE.EXE** permite obținerea setului de date intrare-ieșire prin simulare. Seturile de date obținute sunt salvate de asemenea în fișierul **inties.dat**.

Programul **COMP.EXE** realizează estimarea multivariabilă a parametrilor prin metoda **COMP on-line**. Programul primește ca date de intrare numărul de intrări și ieșiri ale procesului, ordinul mediei alunecătoare, ordinul autoregresiei (fișierul **ordin.dat**) precum și seturile de date intrare-ieșire. Parametrii estimați sunt salvați într-un fișier ce conține numere flotante, **param.dat**.

Programul **GRAFIC1.EXE** permite reprezentarea grafică a parametrilor estimați precum și calculul unor indicatori de calitate ai estimării.

În varianta destinată conducerii directe, în vederea creșterii performanțelor, pachetul de programe este simplificat renunțându-se la programul **HELP.EXE**. În acest caz fișierul **param.dat** conține ultimele 500 de valori estimate ale parametrilor.

Prin modificările aduse, crește viteza de prelucrare a unui set de date intrare-ieșire. De exemplu, pentru un proces cu trei intrări și trei ieșiri, timpul de prelucrare se situează sub 4 ms. În conducerea adaptivă acest lucru este suficient pentru reglarea unor procese având constante de timp mai mari de 1 ms, deoarece estimarea parametrilor se face în funcție de viteza de variație a parametrilor, odată la 5 până la 30 de pași.

- Pachetul EBMUL este stocat pe o discetă de 1.2M și necesită arhitectură configurație minimă a sistemului de calcul. calculator compatibil PC-AT cu o unitate floppy de 1.2M și harddisk. Rezultate suficient de bune se obțin la o frecvență de lucru a sistemului de minim 12 MHz.

Bibliografie

1. Identificarea asistată de calculator a sistemelor - M. Terțigă, P. Stoica, Th. Popescu - Editura Tehnică București 1987.
2. Tehnici de identificare și modelare - O. Preștean, I. Mureșan - Curs lite I.P.T.V. Timișoara.
3. The C Programming Language - B.W. Kernighan, D.M. Ritchie - second edition - Prentice Hall Inc.
4. Structuri de date și tehnici de programare - V. Greșu curs lite I.P.T.V. Timișoara 1987.
5. Sistemul de operare DOS. funcții sistem - ediția II-a Cluj-Napoca 1991.

C U P R I N S

- D1. Controler de linie telefonica
A.Graur, D.Potorac, T.Morosan - Univ."Stefan cel Mare"Suceava
pag. 1
- D2. Echipament pentru transmisiuni de date
A.Graur, D.Potorac, T.Morosanu - Univ."Stefan cel Mare"Suceava
pag. 7
- D3. Interfata specializata pentru achizitii de marimi analogice
A.Graur, D.Potorac, T.Morosanu - Univ."Stefan cel Mare"Suceava
pag. 13
- D4. Algoritmi si circuit dedicat de imbunatatire a eficientei
procesului de corectie in transmisiile sincrone cu pondere
constanta
St.Rezeanu - I.P.Iasi, pag. 29
- D5. Circuit de introducere optima a conectiei de faza aferent
unui sincronizator de bit digital cu urmarirea transitiilor
St.Rezeanu - I.P.Iasi, pag. 25
- D6. Metoda neconventionala de achizitionare a informatiei despre
temperatura
Nic.Galupa, V.Mayaya - I.P.Iasi, pag. 31
- D7. Sinteza automatelor secventiale utilizind arbori multivalenti
L.Festila, S.Hintea - I.P.Cluj-Napoca, pag.37
- D8. Procedeu de sinteza automata a circuitelor logice combina-
tionale multivalente
L.Festila - I.P.Cluj-Napoca, pag.41
- D9. Sisteme secventiale analiza/simulare
Fl.Hoza - I.P.Iasi, pag. 47
- D10. Utilizarea inedita a timerului 18253 in comunicatii seriale
Al.Valachi, D.Olaru - I.P.Iasi, pag. 55
- D11. Simulare grafica a unui planificator de actiuni pentru roboti
S.Ticrea, C.Rusu - ACOMIN, I.P.Cluj-Napoca, pag.61
- D12. Arhitectura biprocesor pentru aplicatii de instrumentatie
St.Poli - I.P.Iasi, pag. 71
- D13. Metode de specificare si verificare a protocoalelor in re-
telele de calculatoare
K.Pusztai, V.Dadarlat - I.P.Cluj-Napoca, pag. 81



- Arhitectura unui sistem multiprocesor de tip cub
 I. Grigoras - I.P. Iasi, pag. 85
- Solutii de gestiune a resurselor in implementarea structu-
 rilor de reglare numerica pe arhitecturi de 8 biti
 Oct. Pastravanu, S. Ifrim - I.P. Iasi, pag. 91
- Manifestarea task-urilor in sistemele de conducere a
 proceselor
 M. Litoiu, - I.P. Bucuresti, INMT, pag. 98
43. Alegerea elementelor de calcul in sistemele de conducere a
 proceselor
 M. Litoiu - I.P. Bucuresti, pag. 104
- Sistem de criptare cu cheie publica prin metoda functiilor
 inversabile de tip "Rucsac"
 T. Madu - I.P. Bucuresti, pag. 110
- Operatii geometrice asupra solidelor modelate prin octrees
 D. Gorgan, D. Rusu - I.P. Cluj-Napoca, pag. 118
- Reprezentarea geometrica a modelelor topologice arborescente
 D. Gorgan, L. Albert, D. Rusu - I.P. Cluj-Napoca, pag. 125
- Realismul vizual al reprezentarilor grafice. Graftali
 I. Parkas - U.T. Timisoara, pag. 130
44. Redarea umbrelor scene grafice 3D modelate prin retele de
 poligoane
 I. Parkas, G. Parkas - U.T. Timisoara, pag. 134
45. Metoda de recunoastere a obiectelor poligonale
 S. Ifrim - I.P. Iasi, pag. 138
46. Sistem de laborator pentru dezvoltarea aplicatiilor de con-
 ducere IBM-PC SPOT
 M. Beretoiu, C. Radu, M. Litoiu - I.P. Bucuresti, pag. 142
47. Sistem pentru analiza programelor de automat programabil
 pe IBM (AP MICRO, AP 101, AP 201)
 C. Bologa, C. Salicianu - I.P. Iasi, pag. 150
48. Sistem pentru programelor pentru automatele programabile pe
 IBM (AP 101)
 C. Bologa, C. Salicianu - I.P. Iasi, pag. 154
49. Proiectarea si simularea circuitelor digitale
 D. Gorgan, L. Beretoiu, D. Rusu
 I. Ieremia - I.P. Cluj-Napoca, pag. 158

- E13. Asamblor pentru sisteme cu transputer T 414/T 800
C.Barsan - I.P.Iasi, pag.
L.Albulet, S.Bejenariu, M.Roman
L.Barsan - I.P.T.Iasi, pag.166
- E14. Sisteme expert in comunicatie cu baze de date
D.Carstoiu - I.P.Bucuresti, pag.174
- E15. Sisteme suport de decizie cu aplicatii in medicina
D.Carstoiu - I.P.Bucuresti, pag.182
- E16. Analiza numerica a cimpului magnetic de excitatie la un
vibrator electrodinamic
I.Vadan, S.Pedre, A.Farcas - I.P.Cluj-Napoca, pag.190
- E17. Metoda de apreciere a comportarii intreruptoarelor de
inalta tensiune la comutatie cu ajutorul calculatorului
electronic
G.Curcenu, Gh.Floriganta - ICEMET Craiova, U.T.Craiova
M.Defta - ICEMET Craiova, pag.200
- E18. Program pentru calculul necesarului de caldura pentru
construcii civile si industriale
Cl.Botez - I.P.Iasi, pag.206
- E19. Sistem informatic pentru urmarirea productiei intr-o sectie
de tesatorie sau finisaj
R.Romanescu, C.Codreanu, C.Moga - IPA Iasi
Adri.Condorovici - I.P.Iasi, pag.210
- E20. Sistem informatic pentru procese tehnologice cu elemente
numeroase si tipodimensiuni putine
Gh.Costescu, C.Georgescu - Univ.Galati
G.Gheorghe, S.Stanila - FEA Bucuresti
Z.Vasiliu - REHEL-PRE Galati, pag.216
- E21. Pachet de programe utilizat pentru estimarea in timp real
proceselor multivariabile
D.I.Chiriac, O.Prostean
B.Lustrea - U.T.Craiova, pag.221